# Commodore WORLD

N.º 1. FEBRERO 1984

275 PTAS.

REGALOS y PREMIOS para todos
Ver Editorial Pág. 4



# DESCRIPCIÓN ALFABÉTICA DE LOS MNEMÓNICOS DEL 6502/6510 (I)

## ADC

*Suma la memoria al acumulador con acarreo*

Operación: A + M + C → A, C

(Ref.: 2.2.1)

N Z C I D V
√ √ √ - - √

| Modo de Direc. | Formato en ensamblador | Código Operan. | Núm. Bytes | Núm. Ciclos |
|---|---|---|---|---|
| Inmediato | ADC →Oper. | 69 | 2 | 2 |
| Pág. Cero | ADC Oper. | 65 | 2 | 3 |
| Pág. Cero, X | ADC Oper., X | 75 | 2 | 4 |
| Absoluto | ADC Oper. | 6D | 3 | 4 |
| Absoluto, X | ADC Oper., X | 7D | 3 | 4* |
| Absoluto, Y | ADC Oper., Y | 79 | 3 | 4* |
| (Indir., X) | ADC (Oper., X) | 61 | 2 | 6 |
| (Indir.), Y | ADC (Oper.), Y | 71 | 2 | 5* |

* Suma 1 si se cambia de página.

## AND

*AND lógico con el acumulador*

Operación: A ∧ M → A

(Ref.: 2.2.3.0)

N Z C I D V
√ √ - - - -

| Modo de Direc. | Formato en ensamblador | Código Operan. | Núm. Bytes | Núm. Ciclos |
|---|---|---|---|---|
| Inmediato | AND →Oper. | 29 | 2 | 2 |
| Pág. Cero | AND Oper. | 25 | 2 | 3 |
| Pág. Cero, X | AND Oper., X | 35 | 2 | 4 |
| Absoluto | AND Oper. | 2D | 3 | 4 |
| Absoluto, X | AND Oper., X | 3D | 3 | 4* |
| Absoluto, Y | AND Oper., Y | 39 | 3 | 4* |
| (Indir., X) | AND (Oper, X) | 21 | 2 | 6 |
| (Indir.), Y | AND (Indir), Y | 31 | 2 | 5 |

* Suma 1 si se cambia de página.

## ASL

*Cambia el bit izquierdo*

Operación: C ← | 7 | 6 | 5 | 4 | 3 | 2 | 1 | Ø | ← Ø

(Ref.: 10.2)

N Z C I D V
√ √ √ - - -

| Modo de Direc. | Formato en ensamblador | Código Operan. | Núm. Bytes | Núm. Ciclos |
|---|---|---|---|---|
| Acumulador | ASL A | 0A | 1 | 2 |
| Pág. Cero | ASL Oper. | 06 | 2 | 5 |
| Pág. Cero, X | ASL Oper., X | 16 | 2 | 6 |
| Absoluto | ASL Oper. | 0E | 3 | 6 |
| Absoluto, X | ASL Oper., X | 1E | 3 | 7 |

## BCC

*Salta si acarreo desactivado*

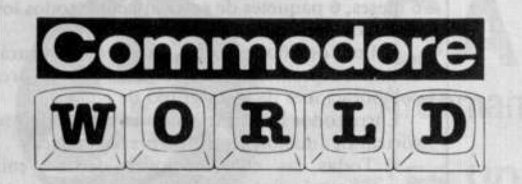
Operación: Salta si C = 0

(Ref.: 4.1.1.3)

N Z C I D V
\- - - - - -

| Modo de Direc. | Formato en ensamblador | Código Operan. | Núm. Bytes | Núm. Ciclos |
|---|---|---|---|---|
| RELATIVO | BCC Oper. | 90 | 2 | 2* |

* Suma 1 si el salto va a la misma página.
* Suma 2 si se salta a otra página.



# SUMARIO 1

Febrero 1984

## CONTENIDO

PAG



## PROXIMO NUMERO

- Disfrutando con las matemáticas (VIC-20 y C-64)
- Conversión del Soft del VIC-20 al C-64
- Ventanas CBM
- Up Periscope para entender valores hexadecimales y binarios
- DISK-O-VIC programa para añadir 13 nuevos mandatos a tu VIC
- Juegos
- Club Commodore todas vuestras aportaciones
- Paquetes de Soft
- Novedades

y... vuestras cartas con comentarios y sugerencias

¡OS ESPERAMOS!


**Commodore World es publicado en colaboración entre Microelectrónica y Control-Commodore y SIMSA**

EQUIPO

Manuel AMADO; Adela LOPEZ; María LOPEZ; Juan MARTINEZ;
Pere MASATS; Jeffrey MILLS; Rafael NAVARRO; Fernando M. RODRIGUEZ;
Manuel SANS; Jordi SASTRE; Valerie SHANKS...
...Y NUESTROS LECTORES

**SIMSA**
**Coordinador María López**
**Pedro Muguruza, 4-8ºB**
**Madrid-16**
**Teléfono (91) 259 54 78**

**MICROELECTRONICA Y CONTROL-COMMODORE**
**Coordinador Pere Masats**
**Taquígrafo Serra, 7-5º**
**Barcelona 29**
**Teléfono (93) 250 51 03/02**

**Imprime: IBERDOS, S.A. Germán Pérez Carrasco, 24. Madrid-27**
**Depósito Légal: M-2944-1984**

# EDITORIAL

# Nueva Etapa para los Lectores Nuevos y Bienvenida a los Viejos

Lo primero de todo —Bienvenidos todos al más amplio mundo de Commodore, *Commodore World*. Especialmente saludamos a los leales del *Club Commodore*.

La revista nuestra y vuestra *Club Commodore* ha crecido tanto, y con tanto éxito, que se hizo imprescindible su mayor profesionalización. Por esta razón, llegó a un acuerdo con la empresa editorial SIMSA para que se hiciera cargo de su publicación.

Esta asociación con SIMSA no va a representar, en ningún momento, un abandono de *Club Commodore* por parte de Microelectrónica y Control. Todo lo contrario.

Microelectrónica continuará escribiendo las secciones familiares a los "viejos" lectores y podrá añadir material nuevo de gran interés.

La sección de Club, como tal, vuestras colaboraciones, sugerencias, preguntas, participación, cartas, se potenciará al máximo con la mayor asistencia técnica.

SIMSA, por su parte, aportará su equipo de profesionales en el periodismo informático y con eso, está dicho todo.

Porque pretendemos que esto sea un auténtico Club y una gran familia, en nuestro equipo editorial (página 3) no se especifican competencias sino **equipo**, porque tanto los colaboradores de Microelectrónica y Control como los de SIMSA van a trabajar juntos en un trabajo de auténtico equipo.

Para lectores "viejos" y nuevos ofrecemos a continuación una breve historia de *Club Commodore* y el nacimiento de su segunda época como *Commodore World*.

### BREVE RESUMEN

En septiembre de 1982, un equipo de profesionales de Microelectrónica y Control —importador exclusivo de los productos Commodore en España— creó el número cero de *Club Commodore* con 8 páginas. Estas 8 páginas se publicaban en el suplemento *Micro Bit* de *Revista Española de Electrónica* y, de forma separada mediante la adición de unas tapas se distribuía a los suscriptores. Pronto se vio la necesidad de aumentar el volumen de la revista y, a partir de diciembre de 1982, *Club Commodore* se convirtió en un adolescente de 16 páginas, con un número de suscriptores en continuo crecimiento y, en abril del 83, apareció por primera vez con 20 páginas. Durante toda la primera época, *Revista Española de Electrónica* ha estado incluyendo en su contenido 8 de las páginas de *Club Commodore*.

Microelectrónica y Control ha pretendido con Club Commodore —como principio— dos cosas que consideramos de gran importancia:

1º Contribuir a la difusión y comprensión de los **ordenadores personales** en nuestro país, manteniendo al usuario informado de las novedades que van apareciendo y avances que se van realizando en este campo y —sobre todo— ayudándole a entender y utilizar más y mejor su equipo, sacándole el "máximo jugo".

2º Servir de medio de comunicación entre los propios usuarios, creando un auténtico **club de amigos de todas las edades**, poniendo nuestras páginas a disposición de **todos** nuestros lectores donde puedan comunicarse entre sí, intercambiando ideas, programas, opiniones, sugerencias y —que incluso— puedan llegar a conocer si lo desean.

**Club Commodore**, una importantísima sección de Commodore World, a partir de hoy, debe ser obra del propio usuario, miembro automático del Club, para ayudar —en la medida de sus fuerzas— a impulsar el avance y educación de la microinformática en España.

Ningún esfuerzo queda perdido —todo colaborador de nuestro Club se convierte automáticamente en alumno y catedrático a la vez—. Desde el programa más técnico al más simple, desde la sugerencia más erudita a la más elemental, todas nuestras colaboraciones son vitales.

La ideología de SIMSA, es exactamente la misma —razón por la que fue elegida por Microelectrónica y Control para celebrar su mayoría de edad de Club Commodore.

### PRESENTE Y FUTURO

Muchos son los proyectos que existen para el **Club Commodore** —al que repetimos, pertenecen todos los orgullosos propietarios de un equipo— en principio explicaremos que ya existe con las adiciones correspondientes al integrarse en Commodore World.

En segundo lugar, os daremos una idea de algunos de los proyectos que tenemos "in mente" para el futuro, pero no un futuro a largo plazo sino a uno que se halla a la vuelta de la esquina, un futuro que es casi un ya.

### PRESENTE

Colaboraciones de todo tipo —programas y artículos de interés (ver **intercambiando experiencias**, página 16).

b) Trucos y sugerencias (ver Magia, página 32).

c) MarketClub (página 34).

d) Los programas que se envíen —que deben venir todos con cinta o disco se devolverán lo antes posible, con algún programa de interés.

e) A partir de ahora se organizarán tres grupos de colaboradores:

1.—de 8 a 12 años.
2.—de 13 a 18 años.
3.—de 19 años en adelante.

Para cada grupo de edad, se sortearán cada 6 meses, 6 paquetes de software entre todos los artículos publicados.

A final de año, en fecha que se comunicará, se darán 3 premios especiales a las tres mejores colaboraciones de cada grupo de edad.

Entre todos los contribuidores a "Magia" se sortearán anualmente 3 "sorpresas mágicas".

f) Todas las colaboraciones deben venir escritas a máquina —a doble espacio— y si no es posible así como nuestros colaboradores más jóvenes, pueden enviarlos escritos a mano con **letra muy clara** (¡Chavales para eso sirve cuando vuestro "profe" se enfada porque los deberes son un "churro").

Incluir información detallada del funcionamiento del programa, tanto por lo que respecta a ejemplos de utilización (para comprobar su funcionamiento), como el detalle de lo que hace cada línea o conjunto de líneas.

g) Enviad, de no tener alguna objeción personal, vuestra dirección y/o teléfono para que podáis poneros en contacto unos con otros.

h) Todas las colaboraciones deben enviarse a Pere Masats, Microelectrónica y Control-Commodore, Taquígrafo Serra, 7, 5.ª - BARCELONA-29 antes del 10 de cada mes.

### FUTURO

Desde ya, próximo número de marzo, enviad vuestras cartas con opiniones, dudas, consultas, cosas que queréis ver en la revista a nuestra redacción: Commodore World, María López Morán. Pedro Muguruza, 4-8º B - MADRID-16.

b) Los diversos Clubs de Commodore que existan en el país, enviarnos detalles sobre los mismos a la redacción para poder organizar entre todos alguna "juerga" (De esto hablaremos en próximos números).

c) Bolsa de trabajo.—Para nuestros informáticos que estén buscándolo, gratis. Para las empresas que necesiten personal 300 pesetas por línea (a redacción).

### UN NOMBRE

COMMODORE WORLD fue elegido entre todos nosotros porque significa **EL MUNDO DE COMMODORE** —un mundo internacional y grande—. Tenemos el orgullo de ser la primera revista dedicada a Commodore en Europa (aparte de Inglaterra donde hay unas cuantas) y también la primera de habla española en el mundo. (Nos están llegando pedidos de Sudamérica).

### Y ACABAMOS

No sin antes expresar nuestro agradecimiento a **Revista Española de Electrónica** que tanta ayuda prestó a **Club Commodore** en su primera etapa y a Carlos Domenech, Presidente de Microelectrónica y Control —y por tanto de Commodore en España— por el continuo apoyo y dedicación, tanto en la primera etapa como en la segunda.

Con estas palabras sobre **Club Commodore** y **Commdore World** damos un abrazo fuerte en un brindis alegre a todos vosotros.

*El equipo*

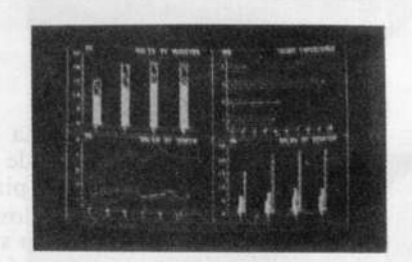
## INDICE DE ANUNCIANTES



# SAKATI, S. A. SOFTWARE PROFESIONAL

Ardemans, 24 - Madrid-28 - Tel. 2567794 - Telex 44222 CICI E

## ¿ES UD. USUARIO DE UN COMMODORE-64...? SI ES ASÍ ENTONCES NUESTROS PROGRAMAS SON PARA UD.



### GRAFICOS-64

Le permite trabajar en modo alta resolución. Traza puntos, líneas, rellena áreas de color, define colores de pantalla y fondo, abre ventanas en la pantalla, escribe y superpone textos en los gráficos, invierte textos. 35 nuevos comandos harán de su trabajo una diversión. MANUAL EN CASTELLANO.

P.V.P. en disco 9.000.— Pts.
P.V.P. en cinta 7.500.— Pts.

### EXTENSION DE BASIC

Potencie el sistema operativo de su COMMODORE-64 con este cartucho que añade 42 nuevos comandos - de gran utilidad. Trabaje con BASIC 4.0 de COMMODORE. Podrá localidar y corregir programas con gran facilidad, pasar programas de un CBM-8000 a un 64, potenciar el editor, y conectar sin ningún interface, sólo con un cable, su 64 a una impresora CENTRONIC. MANUAL EN CASTELLANO.

P.V.P. 25.000.— Pts.

### DESENSAMBLADOR

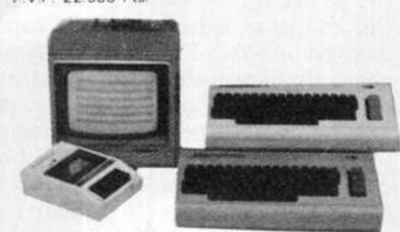
Le permite desensamblar cualquier programa realizado en código máquina. INSTRUCCIONES EN CASTELLANO.

P.V.P. 2.000 Pts. en cinta

### BASE DE DATOS I

Es algo más que un potente generador de aplicaciones. Los programas por aplicación son ilimitados. Aporta un nuevo lenguaje con más de 40 nuevos comandos. Podrá diseñar su propia configuración, con registros de hasta 1000 caracteres y gran facilidad de modificación de los mismos. 127 campos por registro. Contará con 15 ficheros por aplicación ilimitados. Los ficheros de clave y de memoria son así mismo ilimitados. Enlace con EASY SCRIP. Calendario aritmético. Esta Base de Datos es el número uno en su género, algo que Vd. debe tener para su trabajo. MANUAL EN CASTELLANO.

P.V.P. 22.500 Pts.



### BASE DE DATOS II

12 opciones presentadas en el menú. Cree nuevos ficheros de datos. Cuenta Vd. con 12 campos. Esta Base de Datos es ideal para personas que empiezan a introducirse en este método de trabajo.

P.V.P. 4.000 Pts.



### ¿LE GUSTARIA COMPONER MUSICA?

Nuestro programa hará de Vd. un melómano. MUSICA-64 controla tres voces en un rango de 8 octavas cada voz. Sonarán en cualquiera de estos 4 instrumentos: órgano, clavicornio, silofón y trompeta. Podrá introducir 2.000 notas por voz, y una vez compuesta su música, archívela en cinta o en disco. INSTRUCCIONES EN CASTELLANO.

P.V.P. en cinta 4.000 Pts.
P.V.P. en disco 5.000 Pts.

### QUIMICA

Lecciones asequibles de química para sus hijos y para todos los padres de los hijos que estudian química.

P.V.P. 2.000 Pts. en cinta

### VELOS AK-64

Su soporte para programar, capaz de ordenar 500 matrices en 30 segundos.

P.V.P. 2.000 Pts. en cinta

## ¡NO NOS OLVIDAMOS DE LOS NIÑOS NI DE LA TARDE DEL DOMINGO!

### JUEGO DE LOS PALILLOS

Cace los palillos movibles y con joy-stick enciérrelos en el menor área posible, pero ¡ojo! los palillos se mueven en todas direcciones.

### GUERRA DE TANQUES

Prepárese a rechazar un ataque de la brigada de tanques demostrando su puntería.

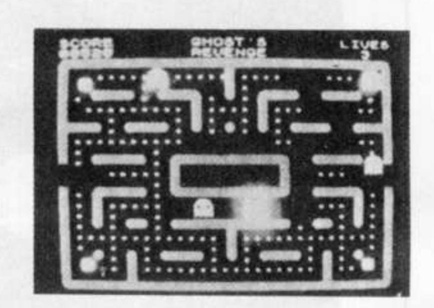
### CACTUS

En su desierto particular, tiene Vd. un hermoso cactus al que amenazan los topos, las águilas y los pájaros.

### COMECOCOS

El popular juego ahora disponible para su 64.

P.V.P. 2.000 Pts. por juego en cinta
P.V.P. 6.000 Pts. por los 4 juegos en disco




## BOLETIN DE PEDIDO

A enviar a: **SAKATI, S. A.** Ardemans, 24. MADRID-28

| Ref. | Cdad. | Precio |
|---|---|---|
| | | |
| | | |
| | | |
| | | |
| | | |
| | **TOTAL** | |

☐ Talon adjunto ☐ Contra Reembolso

Fecha: Firma:

D.

Nº Calle

Código Postal Ciudad

Provincia Teléfono

## ¡TENEMOS ALGO MUY IMPORTANTE! CONTROL DE ALMACEN

Toda aquella empresa que tenga problemas de almacén no debe pararse en este anuncio. Debe llamarnos y le explicaremos con detalle este programa. El espacio no da para más, pero he aquí un avance:

- 1.500 referencias son controladas por almacén, en unidades, litros, kilos, metros, etc., stock mínimo, y lo propio de estos programas, etiquetas, etc.

Pero además el programa tiene una opción de anotación para los gratos de su empresa, con análisis de rentabilidad de su negocio, sin pasar por contabilidad. De manejo sencillo. TODO EN CASTELLANO. El programa se presenta en disco.

P.V.P. 35.000 Pts. Ref. SD14

# Personalizar tu VIC

## con gráficos de juegos

**Ánimo, amantes de los juegos de los ordenadores personales! Incluso el VIC no ampliado puede proporcionar una pantalla de alta resolución y carácteres programables por el usuario para competir con las máquinas de juegos.**

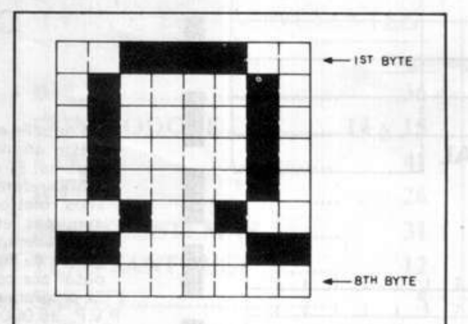
**VIC-20**

Los programas de juegos no son los únicos que se benefician de la capacidad gráfica del VIC. Los matemáticos, estudiantes de idiomas extranjeros y músicos la pueden utilizar para crear carácteres específicamente pertenecientes a su campo. Símbolos como Σ (sigma), Λ (lambda), Ω (ohm), ~ (tilde), ¨ (umlaut), ♩ (cuarto de tono) y 𝄞 (clave de sol) pueden ser creados y utilizados por programadores que antes no se podían expresar dadas las teclas limitadas del VIC.

El Listado 1 presenta una "hoja electrónica" para que experimentes y crees tus propios caracteres. Pero antes de hablar de los programas tenemos que explicar, aunque sea por encima, la naturaleza de los caracteres y la pantalla del VIC.

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

*La Figura 1a. Representación binaria del carácter ohm.*




*La Figura 1b. Representación de pixels del carácter ohm.*

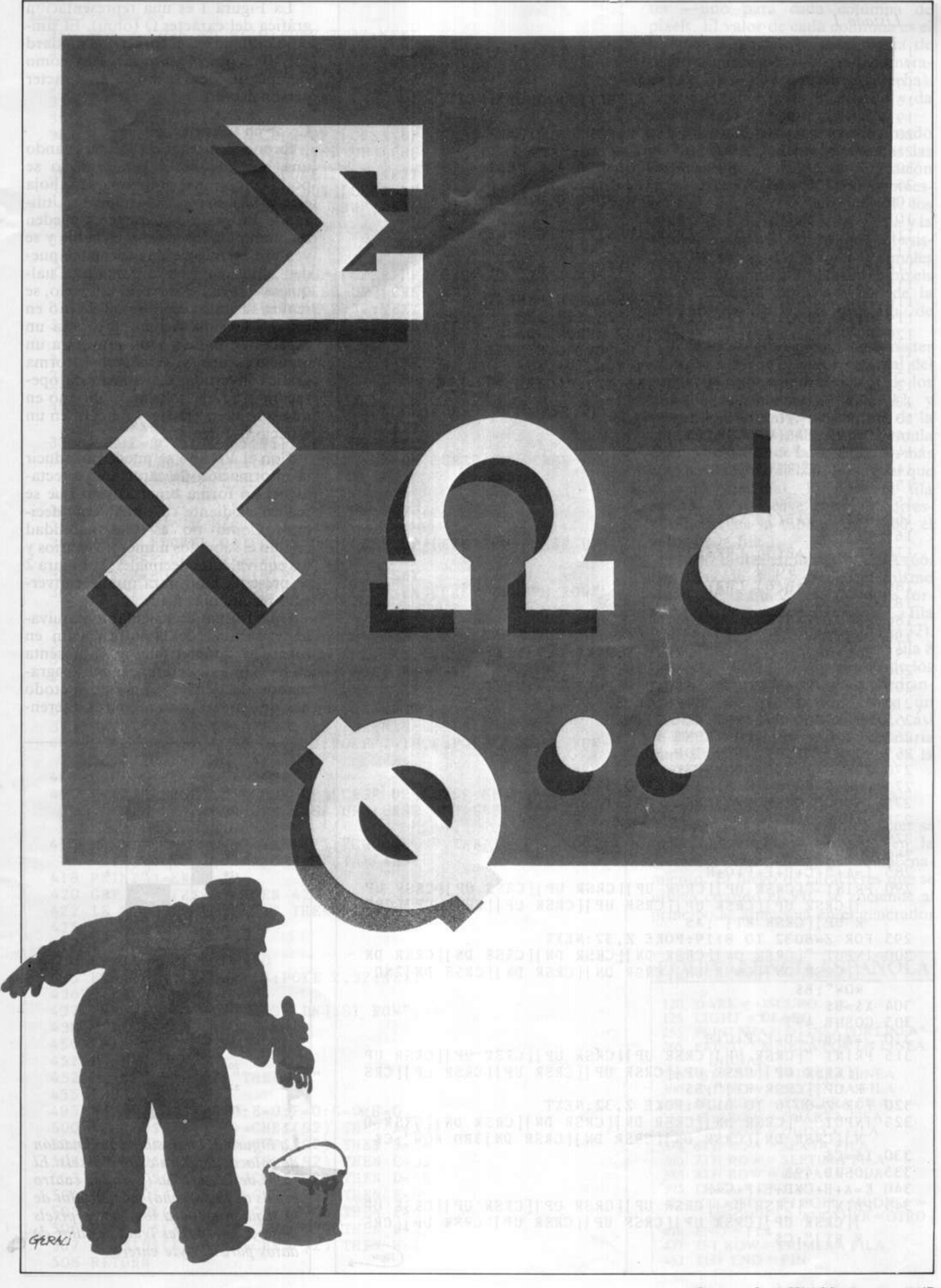
### Caracteres de puntos

Cada carácter en la pantalla del VIC realmente se compone de 64 pequeñas unidades llamadas pixels (puntos). Lo mismo pasa con todos los caracteres, desde la letra A a un símbolo gráfico invertido. Cada carácter tiene una anchura de ocho pixels y una altura de ocho pixels. Estos puntos pueden encenderse (claro) o apagarse (oscuro). La colocación de los pixels oscuros en relación a los pixels claros presenta la ilusión del carácter entero.

La memoria del VIC tiene un método especial de controlar caracteres y de saber cuáles son los pixels que se encuentran encendidos y los que se encuentran apagados. Este proceso se llama "bit mapping" (correlación de bits), en la cual cada pixel se representa por un bit determinado en la memoria.

A lo mejor ya sabes que toda la información almacenada en el ordenador se encuentra en la forma más básica, el lenguaje binario. Los números binarios se componen sólo de los dígitos binarios (bit) 1 y 0. Este sistema es la forma más sencilla de comunicación; una representación numérica de "on" (encendido) y "off" (apagado); sí y no; oscuro y claro. Los bits se agrupan en unidades de ocho que se llaman bytes, en las ubicaciones de memoria del ordenador. La cantidad de memoria que posee el ordenador se expresa en K para kilobytes (1K es igual a 1024 bytes, ó 8192 bits), representando la cantidad de información que es capaz de contener.

Los bits juegan un papel muy importante en la creación de los símbolos en el VIC. Dado que los bits se pueden encender o apagar, se utilizan directamente en la memoria del VIC para representar el estado de cada pixel. Dado que cada posición de memoria almacena un byte, una fila entera de ocho pixels se puede almacenar en una posición de memoria en forma de un número binario. Por lo tanto, se necesitan ocho de estas posiciones para almacenar los bytes para un carácter entero.


GERACI

*Listado 1*

```
1 PRINT "[SHFT CLR]"
2 PRINT"[CTRL 1]"
10 POKE 52,28:POKE 56,28:CLR
15 FOR A=7168 TO 7680:POKE A,PEEK(A+25600):NEXT
19 POKE 36869,255
20 FOR A=7384 TO 7391: READ B:POKE A,B:NEXT
25 DATA 0,0,0,0,0,0,0,0
30 FOR A=7392 TO 7399: READ C:POKE A,C:NEXT
35 DATA 255,255,255,255,255,255,255,255
100 FOR G=7753 TO 7759:READ H:POKE G,H:NEXT
105 DATA 16,18,15,7,18,1,6
110 FOR G=38473 TO 38479:READ H:POKE G,H:NEXT
115 DATA 6,6,6,0,0,0,0
116 FOR G=7456 TO 7463:READ H:POKE G,H:NEXT
117 DATA 255,0,0,0,0,0,0,0
118 FOR G=7464 TO 7471:READ H:POKE G,H:NEXT
119 DATA 255,128,128,128,128,128,128,128
120 PRINT SPC(99);"[CTRL 1][UP ARROW]=DARK"
121 FOR G=7472 TO 7479:READ H:POKE G,H:NEXT
122 DATA 128,128,128,128,128,128,128,128
125 PRINT TAB(5);CHR$(38);TAB(12);"[CTRL 1]]=LIGHT"
130 FOR J=1 TO 8
135 FOR G=1 TO 8
140 PRINT TAB(1) CHR$(37);
145 NEXT G
146 PRINT CHR$(38)
150 NEXT J
155 FOR G=1 TO 4
160 PRINT TAB(1) CHR$(36);
165 NEXT G
170 PRINT TAB(5) CHR$(37);
175 FOR G=1 TO 2
180 PRINT TAB(6) CHR$(36);
185 NEXT G
186 PRINT TAB(8) CHR$(36)
190 PRINT
195 FOR G=7400 TO 7407:READ V:POKE G,V:NEXT
200 DATA 28,16,16,16,16,16,16,28
205 FOR G=7408 TO 7415:READ W:POKE G,W:NEXT
210 DATA 8,20,16,16,56,16,16,82
255 PRINT"DO ONE ROW AT A TIME"
260 FOR G=1 TO 4000:NEXT
265 PRINT"START WITH TOP ROW"
270 FOR G=1 TO 4000:NEXT
271 FOR G=8076 TO 8163:POKE G,32:NEXT
275 INPUT"[CRSR UP][CRSR UP]1ST ROW";A$
278 I=0
279 X$=A$
280 GOSUB 495
285 I=A+B+C+D+E+F+G+H
290 PRINT "[CRSR UP][CRSR UP][CRSR UP][CRSR UP][CRSR UP
    ][CRSR UP][CRSR UP][CRSR UP][CRSR UP][CRSR UP][CRS
    R UP][CRSR RT]";A$
295 FOR Z=8032 TO 8119:POKE Z,32:NEXT
300 INPUT "[CRSR DN][CRSR DN][CRSR DN][CRSR DN][CRSR DN
    ][CRSR DN][CRSR DN][CRSR DN][CRSR DN][CRSR DN]2ND
    ROW";B$
304 X$=B$
305 GOSUB 495
310 J=A+B+C+D+E+F+G+H
315 PRINT "[CRSR UP][CRSR UP][CRSR UP][CRSR UP][CRSR UP
    ][CRSR UP][CRSR UP][CRSR UP][CRSR UP][CRSR UP][CRS
    R UP][CRSR RT]";B$
320 FOR Z=8076 TO 8119:POKE Z,32:NEXT
325 INPUT  "[CRSR DN][CRSR DN][CRSR DN][CRSR DN][CRSR D
    N][CRSR DN][CRSR DN][CRSR DN][CRSR DN]3RD ROW";C$
330 X$=C$
335 GOSUB 495
340 K=A+B+C+D+E+F+G+H
345 PRINT "[CRSR UP][CRSR UP][CRSR UP][CRSR UP][CRSR UP
    ][CRSR UP][CRSR UP][CRSR UP][CRSR UP][CRSR UP][CRS
    R RT]";C$
```

Sigue

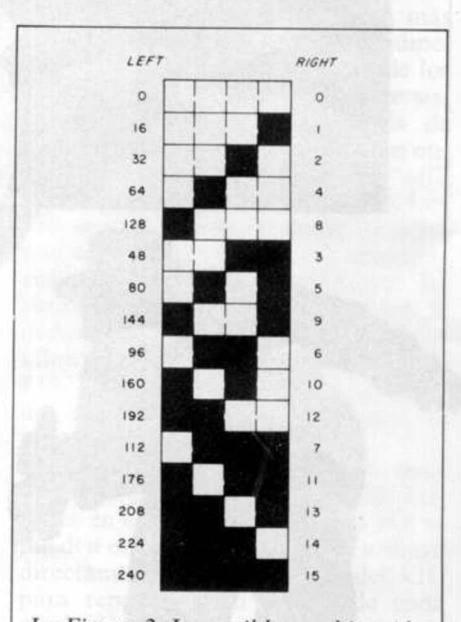
La Figura 1 es una representación gráfica del carácter Ω (ohm). El símbolo ohm no es un carácter estandard del VIC, pero lo utilizaremos como ejemplo de cómo se diseña un carácter personalizado.

**Crear un Carácter**

Se podría diseñar el carácter usando una hoja de papel cuadrado, o se podría emplear el programa de la hoja electrónica Prograf (Listado 1). Utilizando Prograf, los carácteres pueden ser manipulados y desarrollados y se ven en seguida de qué forma se pueden representar en la pantalla. Cualquiera que sea el método utilizado, se realiza la traducción de un diseño en binario escribiendo un cero para un espacio en blanco y un uno para un espacio oscuro. Si se requiere la forma gráfica invertida, se realiza esta operación al revés, colocando un uno en un espacio en blanco y un cero en un espacio oscuro.

Con el VIC no se puede introducir la información de caracteres directamente en forma binaria, sino que se realiza mediante su equivalente decimal. Como no es una habilidad común el saber los números binarios y los equivalentes decimales, la Figura 2 se presenta aquí para que la conversión resulte más fácil.

Una forma de calcular el equivalente decimal de la información en forma de carácter binario se presenta en la "Guía de referencia del Programador del VIC". Con este método hay que sumar ocho números diferen-




*La Figura 2. La posible combinación de colocación para cuatro pixels. El valor de la derecha (para los cuatro pixels de la derecha) más el valor de la izquierda (para los cuatro pixels de la izquierda) es igual al valor de datos para el byte entero.*

*Listado 1 continuación*

```
350 FOR Z=8076 TO 8119:POKE Z,32:NEXT
355 INPUT "[CRSR DN][CRSR DN][CRSR DN][CRSR DN][CRSR DN
    ][CRSR DN][CRSR DN][CRSR DN]4TH ROW";D$
358 X$=D$
359 GOSUB 495
360 L=A+B+C+D+E+F+G+H
361 PRINT "[CRSR UP][CRSR UP][CRSR UP][CRSR UP][CRSR UP
    ][CRSR UP][CRSR UP][CRSR UP][CRSR UP][CRSR RT]";D$
362 FOR Z=8076 TO 8119:POKE Z,32:NEXT
365 INPUT "[CRSR DN][CRSR DN][CRSR DN][CRSR DN][CRSR DN
    ][CRSR DN][CRSR DN]5TH ROW";E$
366 X$=E$
368 GOSUB 495
369 M=A+B+C+D+E+F+G+H
370 PRINT "[CRSR UP][CRSR UP][CRSR UP][CRSR UP][CRSR UP
    ][CRSR UP][CRSR UP][CRSR UP][CRSR RT]";E$
372 FOR Z=8076 TO 8119:POKE Z,32:NEXT
374 INPUT "[CRSR DN][CRSR DN][CRSR DN][CRSR DN][CRSR DN
    ][CRSR DN]6TH ROW";F$
375 X$=F$
376 GOSUB 495
377 N=A+B+C+D+E+F+G+H
378 PRINT "[CRSR UP][CRSR UP][CRSR UP][CRSR UP][CRSR UP
    ][CRSR UP][CRSR UP][CRSR RT]";F$
379 FOR Z=8076 TO 8119:POKE Z,32:NEXT
380 INPUT "[CRSR DN][CRSR DN][CRSR DN][CRSR DN][CRSR DN
    ]7TH ROW";G$
381 X$=G$
382 GOSUB 495
383 O=A+B+C+D+E+F+G+H
384 PRINT "[CRSR UP][CRSR UP][CRSR UP][CRSR UP][CRSR UP
    ][CRSR UP][CRSR RT]";G$
385 FOR Z=8076 TO 8119:POKE Z,32:NEXT
386 INPUT "[CRSR DN][CRSR DN][CRSR DN][CRSR DN]8TH ROW"
    ;H$
387 X$=H$
388 GOSUB 495
389 P=A+B+C+D+E+F+G+H
390 PRINT "[CRSR UP][CRSR UP][CRSR UP][CRSR UP][CRSR UP
    ][CRSR RT]";H$
391 FOR Z=8076 TO 8119:POKE Z,32:NEXT
395 PRINT"[CRSR DN]DATA";I;J;K;L;M;N;O;P
400 POKE 7416,I:POKE 7417,J:POKE 7418,K:POKE 7419,L:POK
    E 7420,M:POKE 7421,N
401 POKE7422,O:POKE 7423,P
405 PRINT"[CRSR UP][CRSR UP][CRSR UP[CRSR UP][CRSR UP]
    [CRSR UP][CRSR UP][CRSR UP][CRSR UP][CRSR UP]"TAB(
    14);CHR$(95)
415 PRINT"[CRSR DN]";TAB(12);"CHOOSE: ",TAB(12);"1=MODI
    FY",TAB(12);"2=ANOTHER",TAB(12);
418 PRINT"3=END"
420 GET Z$:IFZ$="" THEN 420
422 IF Z$>"3" OR Z$<"1" THEN 420
425 IF Z$="1" THEN 435
427 IF Z$="2" THEN 1
430 IF Z$="3" THEN 450
435 FOR Z=8054 TO 8163:POKE Z,32:NEXT
436 CLR
437 INPUT"[CRSR DN][CRSR DN]1ST ROW";A$
438 GOTO 278
450 PRINT"[SHFT CLR]"
451 POKE 646,88
452 PRINT SPC(17)"THE END"
455 END
495 A=0:B=0:C=0:D=0:E=0:F=0:G=0:H=0
500 IF MID$(X$,1,1)=CHR$(92) THEN A=128
501 IF MID$(X$,2,1)=CHR$(92) THEN B=64
502 IF MID$(X$,3,1)=CHR$(92) THEN C=32
503 IF MID$(X$,4,1)=CHR$(92) THEN D=16
504 IF MID$(X$,5,1)=CHR$(92) THEN E=8
505 IF MID$(X$,6,1)=CHR$(92) THEN F=4
506 IF MID$(X$,7,1)=CHR$(92) THEN G=2
507 IF MID$(X$,8,1)=CHR$(92) THEN H=1
508 RETURN
```

tes —uno para cada columna de pixels. El valor de cada columna es el número 2 elevado a la potencia de dicho número de columna (enumeradas de 7 a 0, de izquierda a derecha). Este método resulta incómodo y da más trabajo que el necesario.

El método alternativo presentado en la Figura 2 demuestra todas las posibles combinaciones de colocación para cuatro pixels. La Tabla demuestra los equivalentes decimales de los cuatro pixels a la extrema derecha y la extrema izquierda. Utilizando este sistema, se suman los valores decimales que corresponden a los cuatro pixels de la extrema derecha y los de la extrema izquierda de una fila de caracteres.

Tomando como ejemplo el carácter ohm, se obtiene el valor decimal del primer byte buscando el valor de los bits de la derecha (■■□□), 12, y sumándolo al valor de los bits de la izquierda (□□■■), 48. En la fórmula D+I=P, el valor de la derecha, D, más el valor de la izquierda, I, es igual que el valor decimal, P, para la fila entera. Se sustituye el valor correspondiente de la tabla, 12+48=60, el valor de la fila 1.

Por lo tanto, en la fila 2, 2+64=66. Las filas de 3 a 5 tienen el mismo valor que la fila 2, dado que los formatos de los bits son idénticos. La fila 6 tiene el valor decimal de 36 (4+32). El valor de la fila 7 es de 195 y la fila 8 tiene un valor de 0. Cada uno de los números decimales obtenidos mediante este proceso puede hacer un "Poke" en la memoria del VIC, y así se obtiene la información necesaria para imprimir el carácter ohm en la pantalla.

**Espacio de Memoria**

Antes de almacenar un carácter se tiene que preparar un sitio en la memoria que sirva de almacenamiento. El juego de caracteres que se utiliza cuando el VIC se enciende al principio se almacena en el generador

## EQUIVALENCIA ESPAÑOLA

| | |
|---|---|
| 120 | DARK = OSCURO |
| 125 | LIGHT = CLARO |
| 255 | PRINT "HAZLO LINEA POR LINEA" |
| 265 | PRINT "EMPIEZA POR LA LINEA INICIAL" |
| 275 | 1ST ROW = PRIMERA LINEA |
| 300 | 2ND ROW = SEGUNDA FILA |
| 325 | 3RD ROW = TERCERA FILA |
| 355 | 4TH ROW = CUARTA FILA |
| 365 | 5TH ROW = QUINTA FILA |
| 374 | 6TH ROW = SEXTA FILA |
| 380 | 7TH ROW = SEPTIMA FILA |
| 386 | 8TH ROW = OCTAVA FILA |
| 395 | DATA = DATOS |
| 415 | CHOOSE = ELEGIR, MODIFY = MODIFICAR, ANOTHER = OTRO |
| 418 | END = FIN |
| 437 | 1ST ROW = PRIMERA FILA |
| 452 | THE END = FIN |

de caracteres ROM. El segundo juego, obtenido al pulsar las teclas "SHIFT" y Commodore a la vez, también se almacena en ROM. Dada la misma naturaleza de ROM (Memoria sólo para Lectura), no acepta ningún carácter personalizado: Sin embargo, el área RAM se encuentra totalmente disponible para su uso.

Existen dos posiciones en RAM que funcionan como punteros que le indican al ordenador dónde se consigue la información de caracteres. Estas dos posiciones se pueden modificar para que apunten a un área en la RAM del VIC no ampliado. Así, se puede colocar cualquier carácter en RAM, y el VIC podrá tener acceso.

Surge un problema cuando los punteros de caracteres se modifican para apuntar hacia RAM: se pierde la capacidad de utilizar el juego de caracteres pre-programado del VIC. Pero esto no resulta tan problemático como parece. Sólo se necesitan unas pocas líneas de programa para programar una sección de RAM con la información de caracteres contenida en ROM.

```
FOR X = 7168 TO 7679
POKE X = PEEK (X+25600)
NEXT X
```

Las direcciones de RAM que reciben la transferencia de información de caracteres de representan con X. En este ejemplo, los primeros .5K de RAM se utilizan (7168 a 7679). Se puede tener acceso a más o menos de este área utilizando valores más altos o más bajos para X y cambiando 25600 a un número que, cuando se suma al primer valor de X, es igual a 32768 (7168 + 25600 = 32768), el principio de la ROM de caracteres. La segunda línea del programa hace un "PEEK" de la información de caracteres de ROM y hace un "POKE" en la posición de RAM fijada por X.

Una vez desplazado el juego de caracteres a RAM, ocurre una cosa sorprendente. El cursor desaparece de la pantalla con el resultado de que no se pueden editar los listados de programa. Realmente no afecta la capacidad de editar pero no se distingue muy bien dónde se encuentra el cursor. Si tú sabes que el cursor se encuentra directamente debajo de la R de "READY", que aparece en la pantalla al completarse una acción, se pueden contar el número de movimientos del cursor que se necesitan para llegar a un espacio determinado.

Hay que tener cuidado al elegir la colocación del juego de caracteres en RAM. No se puede utilizar RAM ampliada porque los punteros de chip del VIC no tienen acceso a este área. Existen dos áreas adicionales que no se deben utilizar para almacenar el juego de caracteres: 0 (aquí se almacenan datos que se requieren para que el ordenador emplee Basic) y 4096 (esto es el principio del almacenamiento del programa Basic).

La parte superior del área de RAM donde se almacenan los programas Basic (4096-7679) es la mejor posición para los caracteres. Dado que hay que guardar un poco de este espacio para el programa Basic que utiliza estos caracteres, el juego de caracteres de nuestro ejemplo sólo utilizará los primeros .5K de este área.

Los 3.5K del área de almacenamiento del programa Basic de VIC se llena a partir de varias áreas de esta región a la vez. Las variables, los "arrays", los "strings" y el mismo programa Basic se escriben en las posiciones de la memoria empezando en direcciones distintas dentro de este área. Las instrucciones del programa Basic se escriben de abajo, 4096, hacia arriba. Cualquier dato que se almacena en serie se escribe en la memoria de arriba, 7679, hacia abajo a 7657.

El punto de partida y el punto final de la información de variables y "arrays" dependen del sitio en que termina el programa Basic. Si se colocara el juego de caracteres en RAM 7168-7679, y si se utilizaran los "strings" en un programa, los datos en serie se escribirían por encima de la información de caracteres, la cual se perdería para siempre. Resulta fácil comprender la importancia de proteger el juego de caracteres después de trasladarlo a RAM.

Igual que los punteros que tienen acceso a la información de caracteres, existen punteros que le indican al VIC donde tiene que empezar a escribir los programas y los datos en serie. Estos punteros de "strings" se pueden fijar para apuntar una posición que se encuentra por debajo del juego de caracteres, así protegiendo la información de caracteres ya que no se puede escribir encima de ésta. La Figura 3 presenta un listado de los sitios en RAM donde se pueden fijar los punteros para que incluyan un juego de caracteres incluso más grande.

| Posición inicial para el juego de caracteres | Se hace un "POKE" 52 y 56 con los valores... | Se hace un "POKE" 36869 con los valores... | Memoria asignada para el juego de caracteres |
|---|---|---|---|
| 5120 | 20 | 253 | 2.5K |
| 6144 | 24 | 254 | 1.5K |
| 7168 | 28 | 255 | .5K |

*La Figura 3. Las direcciones en RAM que permiten juegos de caracteres mayores. Se hace un "POKE" en las posiciones determinadas con los valores correctos para iniciar el juego de caracteres en la posición RAM de la columna uno. (NOTA: estos valores sirven para el VIC no ampliado).*

**Caracteres Personalizados**

Para demostrar estas ideas vamos a programar nuestro carácter ohm. Los datos que calculamos antes se colocarán en las posiciones RAM 7168 a 7175. Este área contiene la información para el símbolo @ cuando se acaba de trasladar el juego de caracteres.

```
5 POKE 52, 28: POKE 56, 28; CLR
10 FOR X = 7168 TO 7679: POKE X,
      PEEK
   (X + 25600): NEXT X)
15 POKE 36869, 255
20 FOR L = 7168 TO 7175: READ D:
      POKE L, D: NEXT L
25 DATA 60, 66, 66, 66, 66, 36, 195, 0
```

Las líneas 5-15 reservan la memoria, modifican los punteros de caracteres y trasladan el juego de caracteres. La conversión del símbolo @ en el símbolo de ohm (Ω) tiene lugar en las líneas 20 y 25.

Al modificar los números de posición en la línea 20 y utilizando los datos para distintos caracteres en la línea 25, el juego entero de caracteres puede ser programado de nuevo para que contenga cualquier forma, símbolo o carácter necesario. Aunque se cambie la apariencia de una tecla, esta todavía retiene la definición original. Un signo de más (+) sigue sumando números a pesar de su nueva apariencia.

Antes de modificar un carácter hay que saber las posiciones del principio y del fin de dicho carácter en RAM. Para poder hacer esto, se obtiene el código de pantalla del carácter de la guía del usuario del VIC. Para saber la dirección inicial se multiplica este número por ocho y se suma el resultado al número que representa la posición inicial del juego de caracteres en RAM. Para obtener la dirección final se suma siete a este número.

**Alta Resolución**

El uso de gráficos de alta resolución produce una representación en panta-

lla más clara y más profesional. También exprime la memoria. Cuando hablamos de los caracteres personalizados dijimos que el área más pequeña que se puede manipular en la pantalla del VIC es de un pixel, que representa 1/64 de un carácter entero. Para usar los gráficos de alta resolución, hay que controlar los pixels de cada carácter, y esto se hace mediante el "bit-mapping" como hemos mencionado antes.

La pantalla del VIC tiene 22 caracteres de anchura y 23 de longitud. Esto significa que la pantalla entera del VIC tiene una resolución de 176 × 184 pixels, para un total de 32.384. Cuando cada pixel se asigna a un bit individual, se requiere un total de 4048 bytes para su almacenamiento. Obviamente, esto no es posible en el VIC no ampliado. Sin embargo, es posible realizar el "bit-mapping" en una porción de la pantalla del VIC: Un área de 8 × 8 caracteres (64 × 64 pixels) requiere sólo 512 bytes de memoria para almacenar la información, y esto cae dentro de la capacidad de memoria.

En este ejemplo, queremos combinar caracteres programables y gráficos de alta resolución. Se utilizan todos los pasos detallados antes para definir los caracteres, además de los procedimientos que se requieren para preparar la pantalla para los gráficos de alta resolución. El primer paso es borrar una porción de la memoria de pantalla para usarla para el "bit-mapping". La memoria de pantalla se ubica directamente encima del área de almacenamiento del programa Basic, desde 7680 a 8191. Para borrar los bits aleatorios de este área se emplea este comando.

```
FOR S = 7168 TO 7679: POKE S,O,
    NEXT S
```

Esto hace un "POKE" de los bits de cero en las ubicaciones 7168 y 7679 de la memoria de pantalla. Un repaso rápido de los valores de las direcciones de memoria indica una diferencia de 512. Esta cifra corresponde a los 512 bytes que hemos asignado para la pantalla de alta resolución.

Un programa de alta resolución requiere varias fórmulas para ubicar y controlar los pixels individuales. Se puede comparar esto a escribir una carta a un amigo. Existen varias maneras de escribir la dirección de esa persona —cada una más específica que la anterior. Tienes que saber el país, el estado, la ciudad, el prefijo, el nombre de la calle, y, por último, el nombre de tu amigo.

La localización de un pixel específico se realiza con la misma precisión. El carácter donde "reside el pixel es la unidad más grande donde éste se encuentra, como el país de tu amigo. Esta fórmula calcula el código de representación del carácter dentro de la formación de caracteres de 8 × 8.

C = INT (X/8)*8 + INT (Y/8).

C representa el número de código de representación del carácter, y X e Y son las coordenadas horizontal y vertical del pixel que se va a trasladar.

Se calcula la fila del pixel dentro del carácter mediante la fórmula
R = (Y/8-INT (Y/8))*8.

La variable R es el número de la fila, e Y, de nuevo es la coordenada vertical.

Para calcular el número del byte, o la dirección donde se encuentra el dígito binario del pixel, se multiplica el número del carácter (C) por ocho y se suma el resultado al número de fila (R) y 7168, el comienzo de la memoria de pantalla. (Este número de la memoria de pantalla será diferente si se utiliza otra cosa que no sea los primeros .5K de RAM para el almacenamiento de caracteres). La variable T representa el número del byte.

T = 7168 + 8*C + R

Finalmente, el número del bit se encuentra utilizando la fórmula

I=7-(X-INT (X/8)*8)

El número del bit se representa por I, y las coordenadas horizontal y vertical son X e Y, respectivamente.

**Aplicaciones de Alta Resolución**

Una aplicación ideal del control de pixels de alta resolución es la visualización de ecuaciones matemáticas. Una fórmula para una línea o una curva podría surgir en el programa y sus valores de X e Y podrían ser convertidos por las cuatro fórmulas de control de pixels en los términos necesarios para que la línea o curva apareciera en la pantalla de alta resolución.

De forma semejante, otras fórmulas se podrían emplear para controlar los caracteres de juegos. Cuando esta capacidad para manipular pixels se combina en la pantalla de alta resolución con caracteres especialmente diseñados para las necesidades del programador, las posibilidades no tienen límite.

**Caracteres Multi-colores**

¡El VIC es una cosa maravillosa! Los caracteres normales (al igual que los caracteres programados por el usuario) se componen de dos colores: el color específicamente elegido mediante las teclas de control y de color, y el color de la pantalla. Sin embargo, el VIC tiene una modalidad especial que permite que cuatro colores diferentes se incluyan dentro de un solo carácter. En el VIC no ampliado, esta modalidad puede introducirse haciendo un "POKE en el registro de control de color (646).

Se modifica el valor de este registro cada vez que se cambia el color de un carácter mediante las teclas de control y de color. Si se hace un "POKE" en el registro con un número de cero a siete se da al carácter un color de negro a amarillo, por toda la gama de colores del VIC.

Los cuatro colores que componen cada carácter en la modalidad de multicolor son los de la pantalla actual, el borde y los caracteres, además de un color auxiliar que depende del número que ha hecho el "POKE" en el registro de color.

```
(a) 00 11 11 00        (b) SS AA AA SS
    01 00 00 10            BB SS SS CC
    01 00 00 10            BB SS SS CC
    01 00 00 10            BB SS SS CC
    01 00 00 10            BB SS SS CC
    00 10 01 00            SS CC BB SS
    11 00 00 11            AA SS SS AA
    00 00 00 00            SS SS SS SS
```

| (a) Pares de bits | Asignación de color | Color |
|---|---|---|
| 00SS | pantalla | |
| 01BB | borde | |
| 10CC | carácter | |
| 11AA | auxiliar | |

*La Figura 4. (a) Representación binaria del carácter ohm en pares de bits. (b) Representación en pantalla del carácter ohm en la modalidad de multi-color. (c) Asignación de colores al carácter ohm en la modalidad de multi-color.*

La resolución de la modalidad de multi-color es la mitad de la de la modalidad normal. Es así porque los pixels de la modalidad de multi-color se controlan en pares.

El orden de los dos bits en cada par decide su color. Si el par consiste en dos bits de cero (00), los dos pixels correspondientes tendrán el color de la pantalla. Si el orden de los bits es de 01, el color del borde llenará ese área de pantalla. El orden 10 imprime el color del carácter, y 11 dibuja el color auxiliar. Si colocamos nuestro carácter ohm en la pantalla en la modalidad de multi-color, la distribución de colores sería la que se presenta en la Fig. 4.

El uso de la modalidad de multi-color distorsiona tanto los caracteres que resultan difíciles de reconocer. Sin embargo, teniendo en cuenta el color que cada par representa, sería posible crear un carácter personalizado compuesto de colores en vez de una forma específica. Los bits que se requieren para hacer una tabla de ajedrez de cuatro colores se presentan en la Figura 5.

| Bits | Asignación de color |
|---|---|
| 00 00 10 10 | SS SS CC CC |
| 00 00 10 10 | SS SS CC CC |
| 00 00 10 10 | SS SS CC CC |
| 00 00 10 10 | SS SS CC CC |
| 01 01 11 11 | BB BB AA AA |
| 01 01 11 11 | BB BB AA AA |
| 01 01 11 11 | BB BB AA AA |
| 01 01 11 11 | BB BB AA AA |

*La Figura 5. Orden de bits necesario para crear una tabla de ajedrez de cuatro colores en la modalidad de multi-color.*

AND 128) encuentra la posición de la memoria de colores. Se hace un "DOKE" en la posición 36878 para fijar el color auxiliar. Si se hace un "POKE" de H con un color auxiliar se fija el espacio de la esquina superior de la izquierda en la modalidad de multi-color. Si se añade 1 a H y se hace un "POKE" con el resultado con un color auxiliar el siguiente espacio a la derecha se vuelve multi-color, etc.

La colocación de áreas de pantalla que se fijarán en esta modalidad se elige según el uso dentro del programa. Podría ser que un área determinada de la pantalla en un programa de juego es una zona oculta de puntos adicionales. Cuando un carácter se desplaza hacia este área, se vuelve multi-color, y el jugador sabe que se ha ganado unos puntos adicionales.

Es posible utilizar los gráficos de multi-colores dentro de un programa simplemente añadiendo una línea que hace un "POKE" en la posición 646. Sin embargo, el resultado de esto es que todo lo que se imprima en la pantalla después será de multi-color. Es posible tener mayor control sobre esta modalidad si se establecen los espacios que serán de multi-color. La fórmula H=37888 + 4* (PEEK (36866)

**El programa Prograf**

Prograf es un programa de hoja electrónica que no se desperdicia y que se puede volver a usar. Se elige entre dos teclas que oscurecen (£) o aclaran (() un pixel. Cada una de las ocho filas que componen un carácter programable se colorean por separado. Al completarse las ocho filas, se representa el carácter programable en tamaño real.

Además, se representa en forma de lista la sentencia de Datos correspondiente, que contiene toda la información necesaria para la reproducción del carácter nuevo. Finalmente, un menú proporciona una opción de modificar el carácter existente, construir un carácter totalmente nuevo o terminar el programa.

Prograf se divide en varias secciones diferentes. Esto permite que el programador nuevo siga la construcción de Prograf con facilidad, pero no inhibe su función ni reduce su utilidad. Las líneas 10-19 trasladan los miembros del juego de caracteres a un área reservada de la memoria. En vez de imprimirse, el título principal hace un "POKE" en la pantalla para colorearse a continuación (líneas 100-115).

Los números de carácter 92 y 91 se eligieron para representar un bit "on" (£) y un bit "off" ([). Dos caracteres adicionales (CHR$94 y CHR$ 93) son necesarios para representar las teclas como eran antes para que se representen en un menú. La formación de caracteres de ocho filas se imprime en las líneas 130-190 junto con un menú para las teclas oscuras y claras (las líneas 120 y 125).

Para facilitar la construcción de la formación de caracteres se definieron específicamente tres caracteres adicionales (las líneas 116-122). Cada línea hace una pregunta y se imprime en las líneas 275-390. Una sentencia de Datos, el carácter recién creado y un menú se imprime en la pantalla (líneas 395-415).

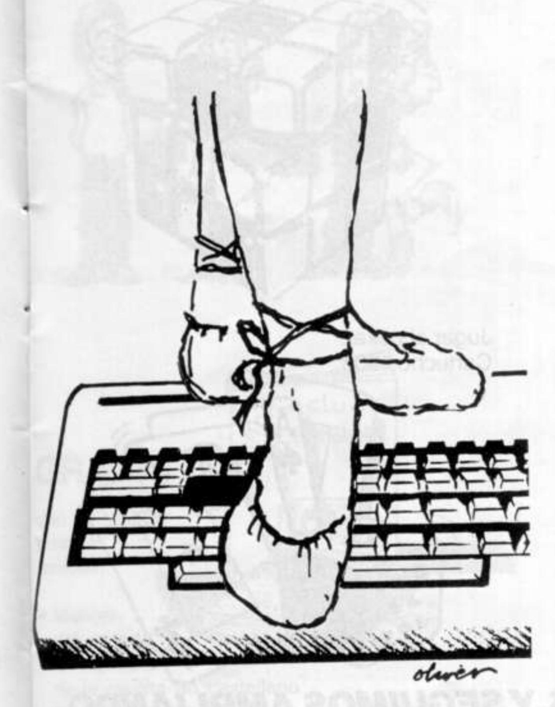
Dado que el menú se motiva mediante una sentencia Get, la información de Datos permanece en pantalla hasta que se realiza una selección del menú. Una opción de "modificar" se remite a las líneas 435-438 para borrar los Datos y borrar las variables de la memoria. Las líneas 450-455 terminan el programa en la extrema modalidad de multi-color. Al final, todos los cálculos para los listados de Datos imprimidos se realizan en Gosub, en las líneas 495-508.

Si se utiliza como es debido, Prograf resulta una herramienta de programación potente que ahorra tiempo. Fue destinado para el uso doméstico de programadores cuidadosos, así que no incluye ninguna provisión de seguridad. La sentencia Get del menú está bien protegido en contra de alguna entrada errónea, pero el resto del programa no está protegido en absoluto. Por lo tanto, si no se pulsa la tecla correcta, si se introducen demasiados pixels o no se introduce una fila completa, se provocará un choque de Prograf.

A veces se puede recuperar totalmente si se pulsan Run/Stop/ y Restore a la vez. Sin embargo, si siguen apareciendo los problemas, hay que comprobar si el listado del programa contiene algún error.

# Seguir el Ritmo del Veloz VIC

**Presentamos unos trucos para reducir la velocidad de los listados del VIC para que sean más fáciles de leer.**




Si vd. toma en serio, o no tan en serio, la programación que hace en el VIC-20 de Commodore, probablemente se habrá sentido frustrado muchas veces. Aunque las limitaciones del VIC son pocas, las que tiene a menudo influyen más que las ventajas que proporciona la máquina.

Uno de los inconvenientes más importantes es el tamaño de la pantalla. Esta sólo contiene 506 caracteres a la vez bajo unas condiciones normales dado que sólo dispone de 23 filas de 22 columnas. Esto da lugar a un problema con el comando de Listado (List). Ya que un listado puede dejar muchas posiciones de la pantalla en blanco, y dado que la mayoría de las líneas en un programa requieren más de una línea de pantalla, no cabe gran cosa en la pantalla en un momento determinado.

Para que el problema sea más acusado, el comando del Listado representa un programa en lo que parece ser una velocidad de la luz. Aún pulsando la tecla de control para reducir la velocidad del listado, sigue saliendo demasiado de prisa para poderlo copiar. La única alternativa parece ser la de presentar unas pocas líneas a la vez.

**Aplicando los Frenos**

Ahora, aquí, y por primera vez, presentamos la solución al problema del "Listado veloz". Este truco no solamente reduce la velocidad del listado; también reduce la velocidad de cualquier otra cosa que realice el VIC, proporcionando unos resultados sorprendentes.

Un aspecto agradable de este truco es que se pueden elegir entre dos velocidades: la de barrido (fallo), y la de copiar (disponible al pulsar el botón). Este procedimiento es muy sencillo. Para reducir la velocidad del VIC, se teclea:

```
POKE 37158,23: POKE 37159,0
```

¡Eso es todo! Ahora al pasar el listado de un programa, adquiere la velocidad de barrido —ni demasiado lento ni demasiado de prisa—. Esta velocidad es la más adecuada para barrer, pero va demasiado rápida para copiar.

Para reducir la velocidad a la de copiar, se pulsa cualquier tecla (excepto la de "restore") o se aprieta la tecla "shift-lock". Esta velocidad le da tiempo de cenar antes de que la pantalla se desplaze a otro juego de caracteres. Para volver a pasar la velocidad de barrido, se sueltan todas las teclas, teniendo cuidado de que la tecla "shift-lock" quede libre.

Para volver a la velocidad normal, se hace una de tres cosas: 1) POKE 37159,66; 2) mantener pulsada la tecla de "run/stop" y pulsar la tecla "restore"; o 3) teclear LOAD.

**Por qué funciona**

La explicación de este truco es que las posiciones de memoria 37158 y 37159 están relacionadas directamente al sistema de interrupción del VIC, el cual controla el reloj interno del ordenador (al que se tiene acceso mediante TI y TI$). Estas posiciones de memoria le indican al VIC la frecuencia con que tiene que actualizar este reloj.

Al reducir el valor de la posición de memoria 37159, el VIC actualiza el reloj con más frecuencia, y por lo tanto, podrá realizar menos tareas en el mismo espacio de tiempo. Cuando tiene un valor de 66, actualiza al ritmo normal. Cuando tiene un valor de cero, actualiza con más frecuencia, y todo lo demás parece ir más despacio.

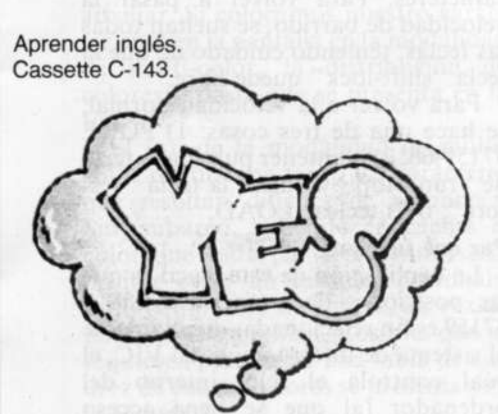
Dado que la velocidad se tiene que reducir más todavía para ajustarse a la velocidad humana (así tenemos una idea de la velocidad que pueden alcanzar los ordenadores y el lenguaje de máquina), hay que afinarlo más aún ajustando la posición de memoria 37158. A mi parecer el valor de 23 funciona mejor que cualquier otro, pero puede que vd. se encuentre más a gusto con otro valor. Hay que tener en cuenta que un incremento o una reducción del valor almacenado en 37158 no significa necesariamente un incremento o una reducción correspondientes de la velocidad del VIC.

Otra cosa para tener en cuenta es que la modificación del valor de la posición de memoria 37159 afecta la velocidad de repetición del cursor. Yo creo que un valor de 30 proporciona la velocidad adecuada para el cursor. También ocurre que esto afecta al reloj del VIC, y por lo tanto, TI$ no calcula bien la hora exacta.

Para terminar, para divertirse un poco (y si tiene la paciencia para hacerlo), ejecute unos programas a estas velocidades reducidas —a veces los resultados son sorprendentes.

# FIJESE EN TODO LO
# AHORA MISMO CON U

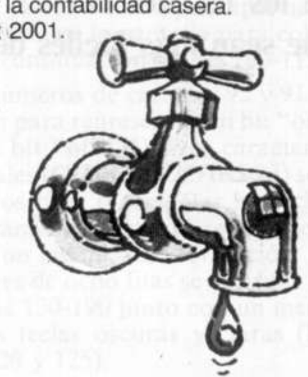
Aprender inglés.
Cassette C-143.



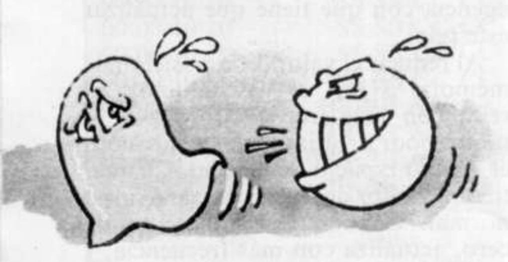
Jugar ajedrez.
Cartucho 1919.



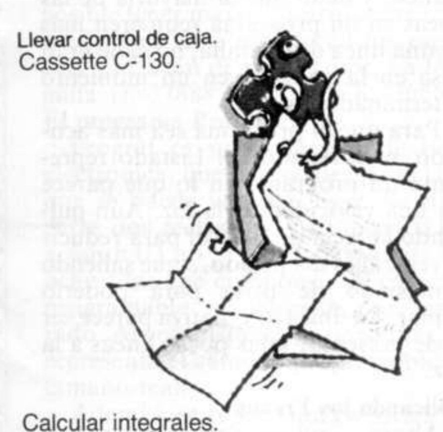
Llevar la contabilidad casera.
Disket 2001.



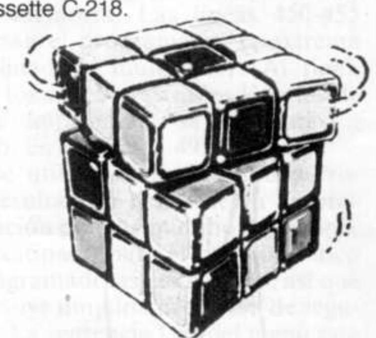
Combatir marcianitos.
Cartucho 1901.



Llevar control de caja.
Cassette C-130.



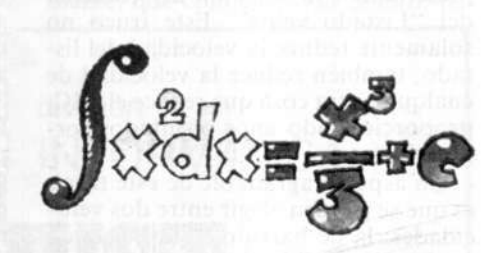
Tocar el piano.
Cassette C-219.




Llevar una agenda profesional.
Disket D-1001.



Calcular integrales.
Cassette C-137.



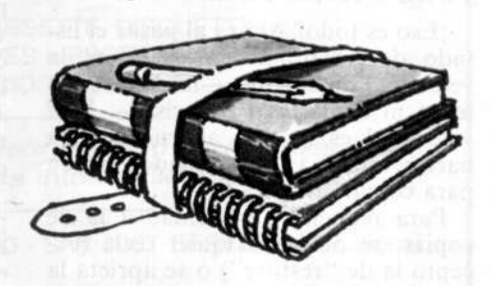

Hacer el cubo de Rubick.
Cassette C-218.



Efectuar tratamiento de textos.
Cartucho C-403.




Preparar tests escolares.
Cassette C-144.

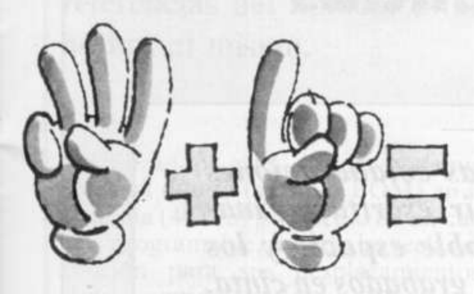


Jugar al póker.
Cartucho 1908.

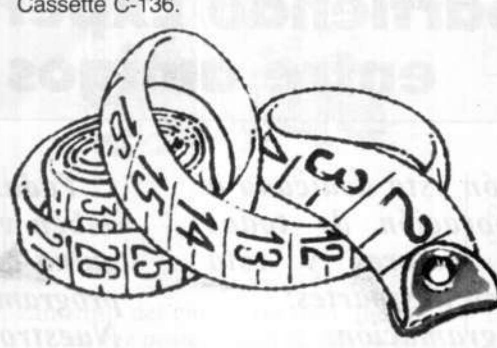
**HASTA 223 CARTUCHOS Y CASSETTES. Y SEGUIMOS AMPLIANDO.**

# QUE PUEDE HACER NA CINTA Y UN BOTÓN.

Enseñar matemáticas a sus hijos.
Cassette C-146.



Programar su dieta ideal.
Cassette C-136.



Saber sus biorritmos.
Cassette C-217.



Aprender el lenguaje del futuro.
Cassette Basic.

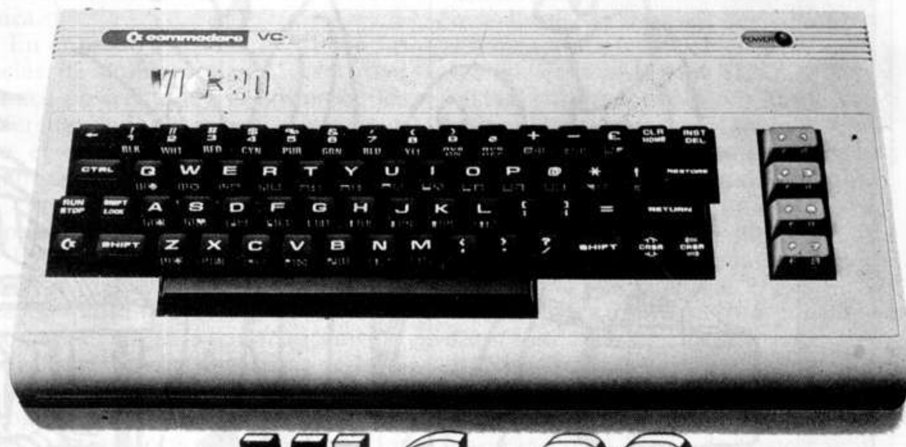


Tener una calculadora HP.
Cassette C-139.



Escoja la cinta de cassette o cartucho que le interese de entre los 223 temas existentes. Colóquelo simplemente en el VIC-20, el ordenador familiar de COMMODORE. Apriete el botón de encendido de su televisor. ¡Y ya está! Siguiendo las instrucciones que aparecerán en la pantalla, usted, su esposa o cualquiera de sus hijos podrá practicar con toda facilidad la actividad que haya seleccionado. Sin problemas. De inmediato. Y de esta forma, al mismo tiempo que puede disfrutar el VIC-20 desde el primer minuto, va introduciéndose de forma sencilla y amena en el mundo de la informática. Leyendo el manual de instrucciones del VIC-20, de regalo, y el curso BASIC, en poco tiempo será capaz de preparar sus propios programas, de inventar sus propios juegos, de buscarle todas las aplicaciones que se le ocurran. Será capaz de dominar el apasionante lenguaje del futuro. Un lenguaje totalmente imprescindible para que sus hijos puedan optar mañana a todo buen puesto de trabajo. Sea cual sea. ¡Prepáreles hoy el camino!

Este es el infinito mundo que el VIC-20 y su amplia gama de periféricos pone a su alcance. El futuro es suyo. Y usted ya sabe que el futuro empieza hov.




**GRATIS**

con su VIC-20 recibirá también:

- Manual del usuario.
- Un ejemplar de la revista en castellano Club Commodore.




## EL ORDENADOR FAMILIAR CON COLOR Y SONIDO.

De venta en tiendas especializadas






# Compartiendo Experiencias entre amigos

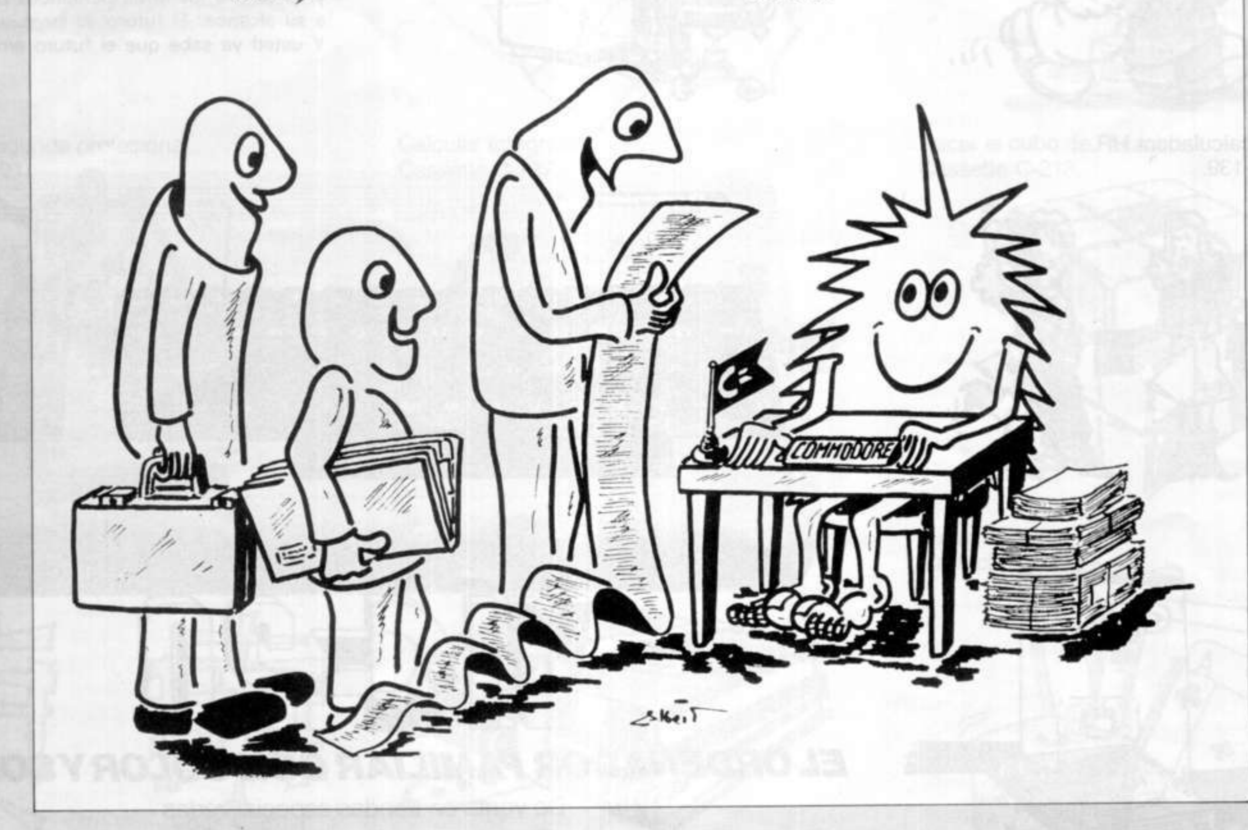
*Esta sección está dedicada a la colaboración de todos nuestros lectores y está dividida en dos partes:*

1) *Programación:* *Programas y similares*
2) *Magia:* *Trucos, sugerencias, etc.*

*Habrá premios y alicientes "para todos los participantes" (ver editorial página 3).*

*Todas las colaboraciones deben venir escritas a máquina a doble espacio y los programas grabados en cinta. Nuestros lectores más jovenes pueden escribir a mano pero con letra muy clara.*

*Enviarnos vuestra dirección para que podáis poneros en contacto unos con otros.*



# Editor de Sprites
## COMMODORE 64



**Se trata de un editor de sprites (o BOM'S). Dispongo únicamente del manual en inglés y por lo tanto mis referencias del manual están hechas al mismo.**




**José Ramón Lasa Urzelai**
**MATXIATEGI, 34-4º A**
**BERGARA**
**Guipúzcoa**
**COMMODORE 64**


Utilizando este editor podemos hacer el dibujo (sprite o BOM) en la pantalla (40×25) por medio de un cursor, programable, usando las teclas de función para sus desplazamientos. Acto seguido podemos acceder a cualquier opción del menú principal como son: GRABAR (un archivo de datos), LEER (un archivo), LISTAR en líneas de basic (de 0 a 60.000) y una vez realizados el/los sprites podemos borrar el programa editor conservando sin embargo todas las líneas de basic (sentencias DATA) obteniéndose la ejecución del mismo, quedando así listo para escribir un programa de juego que utilice sprites, o para grabar estas líneas de sentencias DATA.

En caso de necesidad de modificar el sprite se puede recurrir a la opción 2 del menú principal: CORREGIR.

NOTA: En diversas pruebas que he realizado, no sé debido a qué, una vez borrado el programa (dejando únicamente las sentencias DATA) al intentar cargar las sentencias DATA, previamente grabadas con SAVE...., he obtenido resultados negativos en algunas ocasiones. Tras la desconexión del C64 lo he intentado de nuevo y en este caso sí lo he conseguido.

OTRA nota: El programa está escrito en un TV de B/N y por esto que las referencias son BLANCO y NEGRO: Fondo blanco; fondo negro, línea blanca,... etc.

Después de hacer RUN aparece el menú principal con sus 7 opciones.

Pulsando 1 aparecerá en la pantalla la pregunta: Fondo blanco o negro?

Una vez respondida, el programa dibuja en pantalla el tablero (en blanco o negro) y pasa al menú secundario con otras siete opciones.

Inicialmente pasa al modo ZONA NEGRA y queda a la espera de las acciones del cursor o modos del mismo.

En el modo ZONA (b/n) el desplazamiento del cursor no deja huella.

Una vez posicionado el mismo en la posición deseada se pulsa la barra espaciadora y acto seguido (en la zona de diálogo siempre) se nos pregunta por las coordenadas horizontal y vertical respectivamente. Previa respuesta, el cursor va escribiendo un rectángulo (o cuadrado) cuyo vértice superior izquierdo coincide con la posición del cursor en el momento de pulsar la barra espaciadora. Dicho de otra forma el cursor se mueve hacia abajo y hacia la derecha según los desplazamientos fijados y escribiendo.

En el modo LINEA (b/n) el cursor en su desplazamiento con las teclas correspondientes va imprimiendo una línea.

En el modo PUNTO (b/n) con las teclas de desplazamiento del cursor vamos posicionando el mismo en los diferentes puntos que nos interesen. En el momento que deseemos que aparezca el punto donde está situado el cursor deberemos pulsar la barra espaciadora.

Todo esto en lo referente a las acciones del cursor. Si en su lugar, o sea, si pulsamos cualquier tecla de modo éste cambiará automáticamente. Estas son las teclas numéricas de 1 a 6.

Si pulsamos el asterisco pasa el programa e ejecutar una subrutina que va transformando en bytes los contenidos de las posiciones del tablero correspondientes para acto seguido pasar al menú principal previa presentación en la pantalla del sprite dibujado usando el sprite nº 3 y expandido en los dos ejes de coordenadas.

Si en el menú principal elegimos la opción 2 primeramente se dibujará el tablero y acto seguido el sprite (pero en este caso en la pantalla con definición de carácter, es decir 40×25) que en ese momento esté contenido en la variable B(1) a B(63), bien sea como resultado de un dibujo anterior o como una lectura de un archivo en la cinta del cassette.

En la opción nº 3 podemos crear un archivo en la cinta del cassette.

El programa nos preguntará el nombre del sprite a archivar y ejecutará la acción correspondiente.

Con la opción nº 4 del menú principal podemos recuperar los datos de un sprite previamente archivado en el cassette. Seguidamente podemos corregirlo o listarlo en líneas de BASIC, es decir cualquier acción del menú principal.

En la opción nº 5 borramos el programa dejando en la memoria únicamente las líneas de basic que hayamos listado con la opción nº 6 o cualquier otra línea (de BASIC) cuyo número sea menor que 60.000. Es imprescindible que en el programa exista la línea 60.000 con un REM o con lo que sea para que trabaje perfectamente esta opción si no deberíamos cambiar en la subrutina de borrado por el número de línea a partir de la cual queremos borrar. Los punteros de comienzo de variables, matrices, etc., igualmente se actualizan.

Con la opción número 6 podemos listar los datos del sprite en sentencias BASIC. Nos preguntará por el número de línea inicial y acto seguido pokeando en el buffer del teclado pasarán como líneas de basic con el número inicial dado por nosotros y los seis consecutivos.

En la opción nº 7 se hace END. En el caso de salir del programa por error o por parada voluntaria (tecla de stop) hay que tener en cuenta que la dimensión del buffer de teclado posición 649 queda en 1 y no en 10. Esto se debe a que al inicio del programa se limita para evitar que el cursor se nos



escape al darle repetidas veces a las teclas correspondientes ya que su desplazamiento resulta más lento que una repetición continua de las teclas.

Por lo tanto si queremos recuperar las condiciones iniciales deberemos hacer: poke649,10 RETURN.

Los desplazamientos del cursor se hacen con las teclas de función F1-F7 de la siguiente forma:



| | | |
|---|---|---|
| F1 | CURSOR | SUBE |
| F3 | " | BAJA |
| F5 | " | IZQUIERDA |
| F7 | " | DERECHA |

**MENU PRINCIPAL** (60.200)

Imprime en la pantalla las siete opciones de las que consta.

Hace la variable Y=7 y pasa a la subrutina RESPUESTA A MENU. De la que retorna con la tecla pulsada en la variable X. En función de ese valor se irá a la subrutina correspondiente (ON×GOSUB...)

**MENU SECUNDARIO** (60.400)

Como antes, presenta en pantalla las opciones (7).

Pasa a la subrutina BORRADO de la ZONA DE DIALOGO. Retorna si pulsamos el asterisco (R=42). Si el retorno se ha hecho de otra subrutina por haber pulsado alguna tecla numérica (1 a 6) en la sentencia ON (R-48) GOSUB..., saltará a la subrutina del modo de cursor elegido.

**AUTOMATICO** (60.600)

Va a la subrutina CURSOR y espera allí las órdenes. Cuando retorna de la misma esta nos devuelve la variable W en las siguientes condiciones:

W=2: retorna (cambio de modo de cursor)

W=0: MOVER CURSOR y vuelve al comienzo (60.600)

W=1: cursor en posición. Comienza a ir imprimiendo en la pantalla:

BORRA LA ZONA DE DIALOGO

INPUT coordenadas. Las suma a las del cursor (desplazamiento) y acto seguido las limita a las dimensiones del cuadro.

Bucle FOR-NEXT para ir imprimiendo los puntos ESCRIBE PUNTO.

RETURN.

Esta subrutina maneja las opciones del menú secundario de zonas (ZONA BLANCA, ZONA NEGRA).

**MANUAL** (60.800)

Esta subrutina maneja las opciones: LINEA y PUNTO blanco y negro para ambas. Según los valores de la variable E que codifica el modo del cursor (E=0 será el modo PUNTOS y E=1 LINEAS) y los valores de W (diferentes teclas pulsadas) se ejecutan las acciones correspondientes.

**CURSOR** (61.000)

Halla la posición del cursor, variable S, siendo este valor el de la dirección correspondiente a la memoria pantalla.

Salta a la subrutina de PARPADEO DE CURSOR y después lee el teclado (GET) y según el valor de la tecla pulsada si la hay, da el valor correpondiente a la variable W de la siguiente forma:

Barra espaciadora: R=32 - W=1 y retorna.

Asterisco: R=42 - W=2



Número de 0 a 7: W=1

Si no hay ninguna: vuelve al comienzo de la subrutina.

**MOVER CURSOR** (61.100)

Según el valor de R (tecla numérica pulsada) se hace un salto ON(r-132) GOSUB... y pasa a la subrutina correspondiente: subir cursor, bajar, izquierda o derecha, actualizando en cada caso el valor de la variable en juego (L para desplazamientos verticales y C para los horizontales). Antes de retornar de cualquiera de ellas se limita los valores de las variables L y C a los límites de la matriz de pantalla.

**SUBRUTINAS** (61.800 a 62.090)

Codifican las variables T y E según los diferentes modos del cursor.

T=0 CURSOR NEGRO

T=1 CURSOR BLANCO

E=0 El cursor NO ESCRIBE al desplazarse.

E=1 El cursor ESCRIBE al desplazarse.

**TRANSFORMAR EN BYTES** (62.400)

Esta subrutina va leyendo por la sentencia PEEK todas las posiciones de la matriz de pantalla correspondientes al sprite (24×21); y las va agrupando haciendo el cálculo de los bytes que luego pasarán a sentencias DATA. Estos 63 bytes de que consta el sprite al terminar la subrutina estarán contenidos en la variable B(1) a B(63). Un punto negro corresponderá a un cero y un blanco a un 1 en la lógica binaria.

VARIABLES:

L=Número de línea de la matriz 24×21

V=Líneas completas (memoria pantalla)

O=Líneas completas (Nº de BYTE)

J=Nº de BYTE dentro de la línea actual.

U=Nº de BYTES actuales completos (dir. pantalla).

Q=Nº de BYTE actual.

```
PROGRAMA: EDITOR DE SPRITES


60000 REM  *************************
60010 REM  *                       *
60020 REM  *  EDITOR  DE SPRITES   *
60030 REM  *                       *
60040 REM  *  AUTOR:               *
60050 REM  *  JOSE RAMON LASA      *
60060 REM  *  AGOSTO DE 1983       *
60070 REM- *                       *
60080 REM  *************************
60090 :
60100 REM INICIALIZAR
60110 DIMB(63)
60120 D=1024 :REM MEMORIA PANTALLA CAR.
60130 G=55296:REM  "[2SPC]"   "[3SPC]"
  COL.
60140 F=832  :REM ZONA 13 MEMEORIA
60150 M=53248:REM MAPA DE REGISTRO SPR.
60190 POKE M+21,0
60200 REM MENU PRINCIPAL
60205 :
60210 PRINT"[1CLR][5CRSRD][9CRSRR][1RVSO
N]TABLERO[1SPC]DE[1SPC]SPRITES[1RVSOF]"
60220 PRINT"[3CRSRD][6CRSRR][1RVSON]1[1R
VSOF].DIBUJAR[1SPC]SPRITE"
60230 PRINT"[1CRSRD][6CRSRR][1RVSON]2[1R
VSOF].CORREGIR"
60240 PRINT"[1CRSRD][6SPC][1RVSON]3[1RVS
OF].GRABAR"
60250 PRINT"[1CRSRD][6SPC][1RVSON]4[1RVS
OF].LEER[1SPC]CINTA".
60260 PRINT"[1CRSRD][6SPC][1RVSON]5[1RVS
OF].BORRAR[1SPC]PROGRAMA"
60270 PRINT"[1CRSRD][6SPC][1RVSON]6[1RVS
OF].LISTAR[1SPC]EN[1SPC]BASIC"
60275 PRINT"[1CRSRD][6SPC][1RVSON]7[1RVS
OF].FINALIZAR"
60280 Y=7:GOSUB 62300
60285 IF X=6 GOTO 63300
60290 IF X=7 THEN POKE649,10:END
60300 ON X GOSUB 62600,62800,62900,63100
,63600
60310 GOTO 60210
60320 :
60400 REM MENU SEC.
60405 PRINT"[1HOME][3CRSRD]"
60410 PRINTTAB(28);"[1RVSON]1[1RVSOF].ZO
NA[2SPC][1RVSON]B[1RVSOF][1CRSRD]"
60420 PRINTTAB(28);"[1RVSON]2[1RVSOF].ZO
NA[2SPC][1RVSON]N[1RVSOF][1CRSRD]"
60430 PRINTTAB(28);"[1RVSON]3[1RVSOF].LI
NEA[1SPC][1RVSON]B[1RVSOF][1CRSRD]"
60440 PRINTTAB(28);"[1RVSON]4[1RVSOF].LI
NEA[1SPC][1RVSON]N[1RVSOF][1CRSRD]"
60450 PRINTTAB(28);"[1RVSON]5[1RVSOF].PU
NTO[1SPC][1RVSON]B[1RVSOF][1CRSRD]"
60460 PRINTTAB(28);"[1RVSON]6[1RVSOF].PU
NTO[1SPC][1RVSON]N[1RVSOF][1CRSRD]"
60470 PRINTTAB(28);"[1RVSON]*[1RVSOF].ME
NU."
60475 GOSUB 61700
60480 IF R=42 THEN GOSUB 62400:RETURN
60490 ON (R-48) GOSUB 61800,61870,61920,
61980,62040,62100
60500 GOTO 60475
60510 :
60520 :
60600 REM AUTOMATICO
60610 :
60620 :
60625 RO=R
60650 GOSUB 61000
60660 IF W=1 THEN RETURN
60670 IF W=0 THEN GOSUB 61100:GOTO 60650

60680 GOSUB 61700:POKES+D,105:POKES+G,4
60690 INPUT"[3SPC]HORIZONTAL";H:H=H-1
60700 INPUT"[3SPC]VERTICAL[2SPC]";V:V=V-
1
60710 Y=L:Z=C:C=C+H:X=L:L=L+V:GOSUB 6139
5:K=C:J=L
60720 FOR C=Z TO K:L=Y
60730 FOR L=X TO J
60740 GOSUB 61500
60750 NEXT L,C:L=L-1:C=C-1
60760 R=RO:RETURN
60790 :
60795 :
60800 REM MANUAL
60810 :
60820 :
60870 GOSUB 61000
60880 IF W=1 THEN RETURN
60890 IF W=0 AND E=1 THEN GOSUB 61500:GO
SUB 61100
60895 IF W=0 AND E=0 THEN GOSUB 61100
60900 IF W=2 AND E=1 THEN GOTO 60870
60910 IF W=2 AND E=0 THEN GOSUB 61500
60920 GOTO 60870
60930 :
60940 :
61000 REM CURSOR
61005 ZZ=T
61010 S=L*40+C+2:A=PEEK(S+D):W=0
61020 GOSUB 62200
61040 GET C$: IF C$=""THEN 61020
61050 R=ASC(C$):T=INT(A/160):GOSUB61500:
T=ZZ
61060 IF R=42 THEN W=1:RETURN
61070 IF R=32 THEN W=2:RETURN
61080 IF R>48 AND R<55 THEN W=1:RETURN
61090 IF R>136 OR R<133 THEN 61020
61095 RETURN
61097 :
61098 :
61100 REM MOVER CURSOR
```



B(Q)=Valor del BYTE actual. Inicialmente se pone a cero.
K=Nº de BIT (0 a 7) actual dentro del BYTE.
PEEK(U+K)=Lee el contenido de la posición de pantalla (BIT), si es blanco (160) B(Q)=B(Q)+2 (7-K) actualiza el valor.

**DIBUJAR EL TABLERO (62600)**

Si entramos en esta subrutina desde la opción de "CORREGIR" retorna antes de pasar al menú secundario, debido a que el valor de la variable RE en este caso es 1.

Si la entrada se hace desde la opción 1 del menú de los valores adecuados a L y a C para que el cursor aparezca en el margen superior izquierdo y asimismo el valor adecuado a R para que al saltar al menú secundario pase al modo de cursor zona negra.

**CORREGIR EL SPRITE** (62.800)

Esta subrutina nos hace la función inversa que la de transformar BYTES, es decir: a partir de la variable B(1) a B(63) va pokeando todas las posiciones de la pantalla correspondientes a la matriz 24×21 con los valores (blanco=160 o negro=32) obtenidos de la descomposición en bit's de todos los BYTES.

VARIABLES
I=Nº de línea.
J=Nº de BYTE dentro de la línea.
K=Nº de BIT dentro del BYTE actual.
DB=Valor del byte. Inicialmente DB=B(actual)
PT=Posición en la matriz (dir pantalla)
PI=B/N (punto)

**GRABAR** (62.900); **CONTINUA CASSETTE** (63.000); **LEER CINTA** (63.100); **LISTAR EN BASIC** (63.200)

Pregunta por el nombre del sprite

```
61110 ON (R-132) GOSUB 61200,61250,61300
,61350
61150 RETURN
61190 :
61195 :
61200 REM CURSOR SUBE
61220 L=L-1
61230 GOTO 61400
61240 :
61250 REM CURSOR BAJA
61260 L=L+1
61270 GOTO 61400
61280 :
61290 :
61300 REM CURSOR IZDA.
61310 C=C-1
61320 GOTO 61400
61330 :
61350 REM CURSOR DCHA.
61360 C=C+1
61370 GOTO 61400
61375 :
61395 REM LIMITES DEL CURSOR
61400 IF L>21 THEN L=21
61410 IF L<1  THEN L=1
61420 IF C>23 THEN C=23
61430 IF C<0  THEN C=0
61450 RETURN
61500 REM ESCRIBE PUNTO
61510 S=L*40+C+2
61520 POKE(S+D),(32+128*T):POKE(S+G),14
61530 RETURN
61540 :
61550 :
61700 REM BORRADO Z. DIALOGO
61710 FOR X=1904 TO 2023
61720 POKE X,32: NEXT X
61730 PRINT"[1HOME][21CRSRD]"
61740 RETURN
61790 :
61795 :
61800 REM ZONA BLANCA
61810 PRINT"[1CRSRD][4SPC][1RVSON]ZONA[1
SPC]BLANCA[1RVSOF]"
61820 T=1:E=1
61830 GOSUB 60600
61840 RETURN
61850 :
61860 REM ZONA NEGRA
61870 PRINT"[1CRSRD][4SPC][1RVSON]ZONA[1
SPC]NEGRA[1RVSOF]"
61880 T=0:E=1
61890 GOSUB 60600
61895 RETURN
61900 :
61910 REM LINEA BLANCA
61920 PRINT"[1CRSRD][3SPC][1RVSON]LINEA[
1SPC]BLANCA[1RVSOF]"
61930 E=1:T=1
61940 GOSUB 60800
61950 RETURN
61960 :
61970 REM LINEA NEGRA
61980 PRINT"[1CRSRD][3SPC][1RVSON]LINEA[
1SPC]NEGRA[1RVSOF]"
61990 E=1:T=0
62000 GOSUB 60800
62010 RETURN
62020 :
62030 REM PUNTO BLANCO
62040 PRINT"[1CRSRD][3SPC][1RVSON]PUNTO[
1SPC]BLANCO[1RVSOF]"
62050 E=0:T=1
62060 GOSUB 60800
62070 RETURN
62080 :
62090 REM PUNTO NEGRO
62100 PRINT"[1CRSRD][3SPC][1RVSON]PUNTO[
1SPC]NEGRO[1RVSOF]"
62110 E=0:T=0
62120 GOSUB 60800
62130 RETURN
62140 :
62145 :
62200 REM PARPADEO CURSOR
62210 X=PEEK(S+D)
62220 IF X=32  THEN T=1
62230 IF X=160 THEN T=0
62240 GOSUB 61500
62250 RETURN
62290 :
62295 :
62300 REM RESPUESTA A MENU
62310 GET C$:IFC$=""THEN 62310
62320 X=ASC(C$)-48
62330 IF X<1 OR X>Y THEN 62310
62335 POKED+PT,P1:POKEG+PT,4
62340 RETURN
62390 :
62395 :
62400 REM TRANS.BYTES
62410 POKE53280,6:POKE53281,6
62415 PRINT"[1HOME][22CRSRD][3CRSRR]21[1
CRSRD]"
62420 FOR L=1 TO 21:PRINT"[1CRSRU][2CRSR
R]";L
62430 V=40*L+D+2:O=(L-1)*3
62440 FOR J=1 TO 3
62450 U=V+(J-1)*8:Q=O+J:B(Q)=0
62455 FOR K=0 TO 7
62460 IFPEEK(U+K)=32 THEN 62490
62480 B(Q)=B(Q)+2↑(7-K)
62490 NEXT K,J,L
62500 POKE 53280,254:POKE53281,246
62502 POKE2042,13
62504 FORX=0TO62:POKEF+X,B(X+1):NEXT
62506 PRINT"[1CLR][14CRSRD][3CRSRR]ESTE[
1SPC]ES[1SPC]SU[1SPC]SPRITE"
62507 PRINT"[2CRSRD][3CRSRR]PULSE[1SPC]U
NA[1SPC]TECLA[1SPC]PARA[1SPC]CONTINUAR"
```



para luego escribirlo en un REM, y el número de línea inicial.

Escribe 7 sentencias DATA, con sus números de línea consecutivos, y con nueve datos en cada una de ellas. Finalmente escribe RUN.

Sitúa el cursor en la posición HOME, para que al ir pokeando en el buffer de teclado 13 vaya haciendo el mismo efecto que pulsar la tecla RETURN. Esto lo hará nueve veces: 1 REM+7 DATA+1 RUN. En la posición 198 se pokea 9. Nº de caracteres en buffer de teclado. Dirección inicial del buffer de teclado: 631.

**BORRADO DEL PROGRAMA**
(63.600)

En un número anterior de la revista del club aparece el artículo: "Viaje a las profundidades de la zona de programas". Según este formato de las líneas de BASIC usando los LINK'S vamos buscando aquella posición de la memoria a partir de la cual está el programa, o sea buscamos la línea de programa 60.000. Una vez hallada pokeamos 5 posiciones consecutivas (las cinco primeras de la línea BASIC 60.000) con cero. A continuación los punteros: (174-175) Fin de programa, (45-46) comienzo de variables, (47-48) comienzo de arrays, (49-50) fin de arrays los actualizamos a esta posición (AC). Para ello descomponemos el número decimal contenido en AC en dos:

INT (AC/256)
AC-(INT(AC/256))*256.

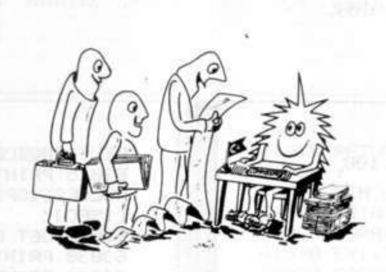
El motivo de haber utilizado unos números de línea tan altos (60.000) se debe a que las sentencias data se pueden numerar en una gama mayor de numeración. En el caso de que no parezca acertada esta elección y optemos por numerar las líneas con cuatro dígitos (o sea no incluir el 6 de izda.) en la línea 63.610 que sería la 3.610 deberíamos poner IF PEEK...= (Número de línea primera del programa).

```
62508 POKEM+21,4:POKEM+4,150:POKEM+5,100
:POKEM+23,4:POKEM+29,4
62515 GET C$:IFC$=""THEN62515
62530 POKEM+21,0:RETURN
62590 :
62595 :
62600 REM DIBUJAR TABLERO
62610 INPUT"[1CLR][10CRSRD][1CRSRR][1RVS
ON]0[1RVSOF].FONDO[1SPC]NEGRO[2SPC][1RVS
ON]1[1RVSOF].FONDO[1SPC]BLANCO";BN:BN=32
+128*BN
62635 PRINT"[1CLR][2SPC]ABCDEFGHIJKLMNOP
QRSTUVWX"
62640 FOR J=1TO 21:L=J*40
62650 PRINTMID$(STR$(J)+"[1SPC]",2,2);:I
FBN=32THEN62690
62660 FOR K=2 TO 25
62670 POKE(D+L+K),BN:POKE(G+L+K),4
62680 NEXT K
62690 PRINT:NEXT J
62695 POKE149,1:IF RE=1THEN RETURN
62710 R=50:L=1:C=0:GOSUB 60400
62730 RETURN
62790 :
62795 :
62800 REM CORREGIR SPRITE
62802 RE=1:BN=32:GOSUB62635:RE=0
62803 PRINT"[1HOME][22CRSRD][3SPC]21[1CR
SRD]"
62805 FOR I=1TO 21:PRINT"[1CRSRU][2CRSRR
]";I
62810 FORJ=1TO3:DB=B((I-1)*3+J)
62815 FORK=0TO7
62820 IFDB-2↑(7-K)>=0THENP1=160:DB=DB-2↑
(7-K):GOTO62830
62825 P1=32
62830 PT=40*I+2+K+(J-1)*8
62840 POKED+PT,P1:POKEG+PT,4
62850 NEXTK,J,I
62860 R=50:L=1:C=0
62870 GOSUB60400
62880 RETURN
62890 :
62895 :
62900 REM GRABAR SPRITE
62910 INPUT"[1CLR][10CRSRD][3CRSRR]NOMBR
E[1SPC]DEL[1SPC]SPRITE";N$
62920 GOSUB 63000
62930 OPEN 1,1,1, N$
62940 FOR K=1 TO 63
62950 PRINT#1, B(K)
62960 NEXT K
62970 CLOSE 1
62980 RETURN
62990 :
62995 :
63000 REM CONT. CASSETTE
63010 PRINT"[1CLR][10CRSRD][3CRSRR]COMPR
UEBA[1SPC]EL[1SPC]CASSETTE":PRINT"[1CRSR
D][3CRSRR](CARA[1SPC]Y[1SPC]NUMERO[1SPC]
DEL[1SPC]CONTADOR)"
63015 PRINT"[2CRSRD][3CRSRR]CUANDO[1SPC]
PULSES[1SPC]UNA[1SPC]TECLA[1SPC]COMIENZO
[1SPC]"
63020 GET C$:IF C$="" THEN 63020
63030 PRINT"[1CLR][12CRSRD][6CRSRR]";
63040 RETURN
63090 :
63095 :
63100 REM LEER CINTA
63110 INPUT"[1CLR][10CRSRD][3CRSRR]NOMBR
E[1SPC]DEL[1SPC]SPRITE";N$
63120 GOSUB 63000
63130 OPEN 1,1,0, N$
63140 FOR K=1 TO 63
63150 INPUT#1, B(K)
63160 NEXT K
63165 CLOSE 1
63170 GOSUB 62500:GOSUB60200
63200 RETURN
63290 :
63295 :
63300 REM LISTAR LINEAS BASIC
63315 INPUT"[1CLR][8CRSRD][3CRSRR]NUMERO
[1SPC]DE[1SPC]LINEA";NL:INPUT"[2CRSRD][3
CRSRR]NOMBRE[1SPC]DEL[1SPC]SPRITE";N$
63320 PRINT"[1CLR][2CRSRD]";
63325 PRINTNL;"REM[1SPC]";N$
63330 FOR J=1 TO 7
63340 PRINT NL+J;"DATA";
63350 FOR K=1 TO9
63360 PRINTB((J-1)*9+K);
63370 IF K<9 THEN PRINT"[1CRSRL],";
63380 NEXT K
63390 PRINT
63400 NEXT J
63405 PRINT"RUN"
63410 PRINT"[1HOME]";
63420 POKE 198,9:POKE649,10
63430 FOR J=631 TO 639
63440 POKE J,13
63450 NEXT J
63500 END
63590 :
63595 :
63600 REM BORRADO DEL PROGRAMA
63605 AC=2049
63610 IFPEEK(AC+2)+PEEK(AC+3)*256=60000T
HEN63700
63620 AC=PEEK(AC)+PEEK(AC+1)*256
63630 GOTO63610
63700 FORX=0TO5:POKEAC+X,0:NEXT
63710 B2=INT(AC/256):B1=AC-B2*256
63720 POKE174,B1:POKE175,B2:REM END PRO
63730 POKE45,PEEK(174):POKE46,PEEK(175)
63740 POKE47,PEEK(174):POKE48,PEEK(175)
63750 POKE49,PEEK(174):POKE50,PEEK(175)
63760 PRINT"[1CLR][10CRSRD][3CRSRR]PROGR
AMA[1SPC]BORRADO[6CRSRD]"
63770 END
```

# 2 PROGRAMAS:
# GRAFICAS VIC 20 + SUPERXPANDER

**El primero de ellos no necesita ningún comentario, es tan sólo una vistosa "introducción" para otros programas que puede ser variado a gusto de consumidor y que demuestra las posibilidades del *Superexpander* en lo que se refiere al dibujo.**


**Guillermo Falgueras Cano**
**Apartado 211**
**LA LINEA**
**(Cádiz)**

El proceso de fabricación no es muy complicado, primero se dibuja en un papel cuadriculado y se divide el papel en 1023×1023 puntos, señalando cada uno en el dibujo, sin olvidar que hay que dividir la componente "Y" del comando DRAW por 0.7 para que el dibujo quede proporcionado. Consume aproximadamente 4 Kb. de memoria.

El segundo programa es más interesante, porque con él podemos estudiar las funciones haciendo su representación gráfica, nos permite ampliar una parte de la gráfica por si nos interesa estudiarla con detalle y nos indica en qué partes de la curva existen asíntotas o lugares en los que la función se hace infinito.

En realidad, representar gráficamente una función no resulta difícil con el **Superexpander**. Debemos definir la función, establecer los valores de la componente "X" y los correspondientes de la "Y". Pero esto a veces, para algunas funciones no resulta tan fácil porque cuando X=0 se hacen infinito y dan error, interrumpiendo el programa, en otros casos esto ocurre cuando X es negativo.

El presente programa trata de evitar todo esto. Al ponerlo en funcionamiento, se nos indica que pulsemos la tecla "F1" y definamos la función a continuación, después pulsemos "RETURN" dos veces y entonces el VIC nos pregunta si deseamos establecer los límites de "X", después de responderle, nos indica que estudia la función, nos da una tabla de valores significativa y nos indica los límites establecidos para los dos ejes así como el valor de cada división del eje vertical. Después es dibujada la gráfica.

Si deseamos ver con detalle una parte de la gráfica sólo tenemos que establecer los valores de la "X" de esa parte concreta.

En general el programa tiene tres partes: A) Definir y estudiar la función (hasta la línea 200). b) Representar la tabla de valores y los límites de los ejes (hasta la línea 300). c) Pintar la gráfica (de la 370 a la 405).

A.—Para definir la función cómodamente defino la tecla "F1" como el comienzo de la línea 80, detrás es definida la función y es entrada en el programa como una cadena "A$" que es igual que una línea de programa. De esta obtenemos una parte "B$" según la línea 50. La línea 65 escribe en la pantalla (en blanco, no se ve) las cadenas A$ y B$ como si fuese una línea de programa y debajo escribe "Run 80". La línea 70 termina el programa, pero detiene el cursor encima del "8" de A$, al pulsar Return una vez se introduce la línea 80 en el programa y el cursor se detiene encima de la "R" de Run 80, y al pulsar Return otra vez continúa el programa.

Las líneas 135-150 estudian la cadena B$ (la función) estableciendo límites de "X" para ciertos tipos de funciones.

Para evitar encontrarnos con error por división por cero y los valores demasiado grandes, se establece la línea 185 en la que a los valores de "X" se les va sumando incrementos (II, IS), si el valor de la función para esos valores es demasiado grande, el siguiente valor es eliminado.

Las líneas 165 y 175 sirve para establecer el valor de K que será el máximo del eje "YY", también para el valor de R que será normalmente uno, salvo cuando el intervalo de XX' que hemos tomado es muy pequeño (R determina los valores en los que aumenta el eje XX')

La línea 195 determina la posición del eje vertical "KV" y la 180 la del horizonte "KH".

B) Las líneas 205 a la 295 estudian y pintan la tabla de valores y los límites de la función. Este estudio de la tabla se hace aparte de la representación.

Las líneas 300 a 365 pintan los ejes, las divisiones de estos y el cuadriculado de la pantalla (este último es opcional por ser útil pero engorroso).

C) La parte más importante del programa esta en las líneas 370 a 405, que son las que pintan la gráfica. La línea 370 prepara el valor de DX, componente horizontal, que depende el primer valor, X1, del número de valores a representar, P, y de la dimensión de la pantalla.

La línea 380 prepara el valor de la componente vertical, DY, que depende de la función, de la posición del eje horizontal, de DX, del nº de valores, adecuándolos a los 1.023 puntos horizontales y verticales de la pantalla.

Según mis observaciones, y después de estudiar cientos de funciones, el programa es capaz de representarlas casi todas, sólo falla en algunos casos

muy especiales que manejan valores demasiado grandes en intervalos muy pequeños, dando entonces error por "OVERFLOW", pero aún en estos casos dibuja parte de la función.

En algunos casos el programa no puede establecer los límites de la "X" por lo que hay que establecerlos cada vez cuando nos pregunta el VIC. Esto ocurre en aquellos casos en los que la función "SOLO" puede tomar valores del eje XX' en intervalos muy cortos (1 a 1 ó 0 a 1 etc.), pero estas funciones también pueden representarse cuando establecemos los límites correctos, y precisamente algunas de ellas son de las más interesantes.

En algunos casos raros la función da algún tipo de error, ello quiere decir que no hemos establecido bien los límites de X, hacemos RUN80 y los volvemos a establecer.

Una misma función se puede estudiar con intervalos diferentes, ello es interesante para algunas funciones.

Este programa consume aproximadamente 6Kb. de memoria. Es interesante entero pero en el caso de ser demasiado largo puede suprimirse la tabla de valores y el cuadriculado de pantalla. A continuación se dan ejemplos de funciones interesantes.

## FUNCIONES UTILES

**A. Que establece sola los límites del eje XX':**

$A(X) + X^3/(X^2-1)$

$A(X) = (X^2+2X)/X^2+2X-8)$

$A(X) = (X+\sin X)$

$A(X) = \mathrm{SIN}(x)\ 3X$

$A(X) = \tan(X) - X$ etc, etc, etc.

$A(X) = 1/(1-X^3)$

$A(X) = 2 \sin(X)$

$A(X) = X \sin(X)$

$A(X) = X^2/(X^2-1)$

**B. Que hay que establecer límites:**

$A(X) = \sin(X)^{\cos(X)}$ entre 0 y 3

$A(X) = \cos(X)^{\sin(X)}$ entre -1.4 y 14

$A(X) = X.\ \mathrm{SQR}(1+X)/1-X))$ entre -0.9 y 0.9

$A(X) = 1/\mathrm{SQR}(1-4X^2)$ entre - 0.4 y 0.4

etc...

```
PROGRAMA: FALGUERAS.GRAFICAS

1 GRAPHIC0:CLR
5 PRINT"[1CLR][3CRSRD][3CRSRR]G.FALGUERA
S[1SPC]CANO":PRINT"[3CRSRD][5CRSRR]APART
ADO[1SPC]211":PRINT"[1CRSRD][5CRSRR](TF.
766999)"
10 PRINT"[3CRSRD][3CRSRR]LA[1SPC]LINEA(C
ADIZ)"
15 PRINT"[5CRSRD][5CRSRR]**********":PRI
NT"[5CRSRR]*PRESENTA*":PRINT"[5CRSRR]***
*******"
20 FORT=1TO4000
25 NEXT
30 POKE36879,25:KEY1,"80DEFFNA(X)="
35 PRINT"[1CLR][3CRSRR][1GRN][1RVSON]GRA
FICAS[1RVSOF][1BLU]"
40 PRINT"[2CRSRD]PULSE[1SPC]'F1'Y[1SPC]D
EFINA[1SPC]LA[1SPC]FUNCION"
45 POKE207,4:INPUTA$
50 B$=RIGHT$(A$,LEN(A$)-12)
55 PRINT"[1CLR][1CRSRD]PULSE[1SPC]RETURN
[1SPC][1RVSON]DOS[1RVSOF][1SPC]VECES"
60 FORQ=1TO22:PRINT"[1SHIFF]";:NEXT
65 PRINT"[1WHT][2CRSRD]";A$;":B$=";CHR$(
34);B$;CHR$(34):PRINT"RUN80"
70 POKE214,4:END
85 II=(1E-6):IS=1-(1E-6):IM=(1E+5):KH=10
23:KV=0
90 PRINT"[1CLR][1BLU]EL[1SPC]PROGRAMA[1S
PC]ESTUDIA[1SPC]LA[1CRSRD]FUNCION[1SPC]Y
[1SPC]ESTABLECE[3SPC][1CRSRD]LOS[1SPC]LI
MITES[1SPC]DE'X'"
95 PRINT"[3CRSRD]?DESEA[1SPC]ESTABLECERL
OS[2SPC]VD.?(S/N)"
100 GETP$:IFP$=""THEN100
105 IFP$="N"THEN125
110 INPUT"[2CRSRD]VALOR[1SPC]MINIMO";X1
115 INPUT"[1CRSRD]VALOR[1SPC]MAXIMO";X2
120 L=1
125 PRINT"[1CLR][1BLU][1CRSRD]ESTUDIANDO
[1SPC]LA[1SPC]FUNCION:[1CRSRD]A(X)=";B$:
PRINT"[2CRSRD][2CRSRR][1RVSON]ESPERE[1SP
C]POR[1SPC]FAVOR[1RVSOF]":IFL=1THEN165
130 FORI=1TOLEN(B$)-4
135 C$=MID$(B$,I,4)
140 IFC$="/LOG"THENX1=1.1:X2=10.1:GOTO16
5
145 IFC$="SQR("ORC$="LOG("THENX1=.1:X2=1
0.1:GOTO165
150 IFC$="EXP("THENX1=-3.9:X2=4.1:GOTO16
5
155 NEXTI
160 X1=-9.9:X2=10.1
165 I=X1:K=ABS(FNA(I)):R=1:IF(X2-X1)=<4T
HENR=.20:II=II*R:IS=IS*R*R
170 IFI=>X2THEN195
175 IFABS(FNA(I))>KTHENK=ABS(FNA(I))
180 IFFNA(I)<0THENKH=512
185 IFABS(FNA(I+II))>IMORABS(FNA(I+IS))>
IMTHENI=I+2*R:GOTO170
190 I=I+R:GOTO170
195 P=X2-X1:KV=ABS(X1)*1023/P:IFX1>0THEN
KV=0
200 IFX2<0THENKV=1023:P=P+ABS(X2)
205 PRINT"[1CLR][4CRSRR][1RVSON]TABLA[1S
PC]DE[1SPC]VALORES[1RVSOF]"
210 PRINT"F(X)=";B$:PRINT"[1CRSRD][2CRSR
R]X[9CRSRR]F(X)"
215 FORI=0TO21:PRINT"[1SHIFC]";:NEXT
220 I=-7.9:IFX1>ITHENI=X1
225 IFFNA(I+II)>=KTHENI=I+R
230 IFCO=>16ORI=>X2THEN250
235 PRINTINT(I*100+.5)/100;"[2CRSRR]";FN
A(I):CO=CO+1
240 IFABS(FNA(I+II))>IMORABS(FNA(I+IS))>
IMTHENI=I+2*R:GOTO225
245 I=I+R:GOTO225
250 PRINT"[4CRSRR][1RVSON]PULSE[1SPC]UNA
[1SPC]TECLA[1RVSOF]"
255 GETP$:IFP$=""THEN255
260 N=-K:IFKH=1023THENN=0
```



```
265 PRINT"[1CLR][7CRSRR][1RVSON]LIMITES[
1RVSOF]"
270 PRINT"[1CRSRD]'X'VARIA[1SPC]ENTRE":P
RINTX1;"[1SPC]Y[1SPC]";X2
275 PRINT"[3CRSRD]'Y'VARIA[1SPC]ENTRE":P
RINTINT(N*1000+.5)/1000;"[1SPC]Y[1SPC]";
INT(K*1000+.5)/1000
280 PRINT"[3CRSRD]CADA[1SPC]PUNTO[1SPC]E
N[1SPC]EL[1SPC]EJE[2SPC]VERTICAL[1SPC]CO
RRESPONDE[1SPC]A":PRINTINT(K/P*1000)/100
0;"UNID."
285 IFP>30ORP<10THENPRINT"[2CRSRD][3CRSR
R][1RVSON]PULSE[1SPC]UNA[1SPC]TECLA[1RVS
OF]":GOTO295
290 PRINT"[2CRSRD][1RVSON]DESEA[1SPC]CUA
DRICULAS(S/N)[1RVSOF]"
295 GETP$:IFP$=""THEN295
300 GRAPHIC2:IFP$="N"ORP>30ORP<10THEN330

305 FORI=0TOP
310 DRAW1,0,D1TO1023,D1
315 DRAW1,D1,0TOD1,1023
320 D1=D1+1023/P
325 NEXT
330 DRAW1,0,KHTO1023,KH
335 DRAW1,0,KH-4TO1023,KH-4
340 DRAW1,KV,0TOKV,1023
```

```
345 FORI=1TOP
350 POINT0,KV,D2
355 POINT0,D2,KH
360 D2=D2+1023/P
365 NEXT
370 DX=1:IFX1>0THENDX=X1*1023/P
375 P=P/1023:K=KH/K
380 DY=KH-FNA(X1+P*DX)*K
385 Z=5:IFABS((KH-FNA(X1+P*(DX+Z))*K)-DY
)>40THENZ=1
390 IFDX=>1023THEN410
395 IFDY>1023ORDY<0THENDX=DX+2*Z:GOTO380

400 POINT1,DX,DY
405 DX=DX+Z:GOTO380
410 GETP$:IFP$=""THEN410
415 GRAPHIC0:CLR
420 PRINT"[1CLR][2CRSRD][2CRSRR]PARA[1SP
C]OTRA[1SPC]FUNCION[3SPC][1CRSRD][11CRSR
R]PULSE'[1RVSON]S[1RVSOF]'"
425 PRINT"[5CRSRD][2CRSRR]PARA[1SPC]LA[1
SPC]MISMA[1SPC]'[1RVSON]N[1RVSOF]'"
430 GETP$:IFP$<>"S"ANDP$<>"N"THEN430
435 IFP$="S"THEN35
440 GOTO80
```

PROGRAMA: FALGUERAS.MAPA

```
5 GRAPHIC2
10 REGION0
15 DRAW1,529,0TO493,64TO422,64TO404,51TO
369,56TO225,53
20 DRAW1,225,53TO207,25TO171,49TO171,67T
O149,64TO113,102TO119,128TO119,153TO136,
192
25 DRAW1,136,192TO133,384TO100,455TO83,5
20TO100,576TO120,576TO99,715TO208,710TO2
52,757
30 DRAW1,252,757TO235,760TO293,822TO307,
795TO311,805TO333,768TO369,768TO373,756T
O466,770
35 DRAW1,466,770TO502,715TO565,665TO612,
552TO593,435TO632,363TO643,356TO643,320
40 DRAW1,643,320TO717,286TO772,217TO761,
128TO816,77TO900,0
45 DRAW1,493,64TO761,128
50 DRAW1,128,215TO171,190TO171,230TO261,
225TO200,448TO216,601TO195,704
55 DRAW1,216,601TO278,614TO344,537TO454,
576TO511,700
60 DRAW1,819,422TO790,475TO822,460TO836,
486TO858,448TO827,435TO819,422
65 DRAW1,858,409TO880,380TO898,422TO858,
409
70 CIRCLE1,766,576,12,12
75 DRAW1,298,614TO297,690TO404,690TO405,
614TO298,614
```

```
80 DRAW1,297,640TO405,640:DRAW1,297,665T
O405,665
85 REGION5:PAINT1,369,625:PAINT1,369,675

90 REGION0:DRAW1,648,1023TO648,896TO693,
832TO1023,832
95 DRAW1,738,920TO805,908TO791,894TO770,
972TO737,920
100 CIRCLE1,844,982,21,21
105 CIRCLE1,742,975,10,10
110 CIRCLE1,690,1000,8,16
115 DRAW1,710,890TO680,924TO685,873TO693
,890
120 DRAW1,928,915TO885,960TO900,975TO922
,915
125 CIRCLE1,960,875,15,9
130 CIRCLE1,311,800,16,16
135 REGION4:PAINT1,682,50
140 PAINT1,130,512
145 REGION2
150 CHAR16,4,"LA[1SPC]LINEA"
155 FORT=1TO1500:NEXT
160 CHAR3,3,"PULSE[1SPC]SHIFT":CHAR4,8,"
Y":CHAR5,5,"RUN/STOP"
165 CHAR19,0,"Y[1SPC]!ESPERE!":DRAW1,0,9
50TO500,950TO500,1023
```




**ELEKTROCOMPUTER** · PRESENTA SUS NUEVOS PRODUCTOS PARA EL VIC-20 Y EL COMMODORE-64. **DATAMASTER 64** Y **CONTROLADOR · C8**, QUE AMPLIAN LAS POSIBILIDADES DE SU ORDENADOR.

DE VENTA EN DISTRIBUIDORES AUTORIZADOS DE TODA ESPAÑA.

***DATAMASTER** 64 _ SOFISTICADA BASE DE DATOS PARA EL C—64. PENSADA PARA TRABAJAR CON LA UNIDAD DE DISCO 1541. SIENDO MUY VERSATIL APROVECHA AL MAXIMO LA CAPACIDAD DE MANIOBRA Y ALMACENAMIENTO. NUMERO DE REGISTROS VARIABLE "EJ. 5000 DE 30 CARACTERES". FORMATEADOS Y COPIAS PROGRAMADAS. SALIDA A IMPRESORA (PARALELO CENTRONICS Y SERIE RS 232) CHEQUEO OPERACIONES DISCO. GARANTIA 3 MESES. MANUAL COMPL. EN CASTELLANO — P.V.P. 11.800' PTAS.

***CONTROLADOR · C8** _ CONTROLADOR DE 8 RELES PARA EL VIC-20 Y EL C-64. DE FORMA MUY SENCILLA PODEMOS HACER HASTA 255 COMBINACIONES ENTRE LOS 8 RELES, CON UN CONSUMO DE 1000 W. A 220 VOLT. CADA UNO. CON LO CUAL PODEMOS ACCIONAR TODO TIPO DE LUCES O MECANISMOS. INSTRUC. INCLUIDAS. 3 MESES GARANTIA — P.V.P. 9.800' PTAS.




**VIA AUGUSTA - 120 - TEF. (93· 2180699 - BARCELONA - 6**



# Carotena

Es un juego provisto de niveles de velocidad y dificultad, no necesita ampliación de memoria pero requiere la utilización del Joystick.



Programa remitido por
Luis Carballo T.
Por favor, manda tu dirección

"Carotena" es una rápida, furiosa y hambrienta serpiente. Ella corre por toda la pantalla con cuidado de no chocar contra los muros eléctricos, devorando su comida favorita: ¡los asteriscos!, provistos de una provitamina especial que desarrolla su crecimiento, y cuantas más coma más grande se hace.

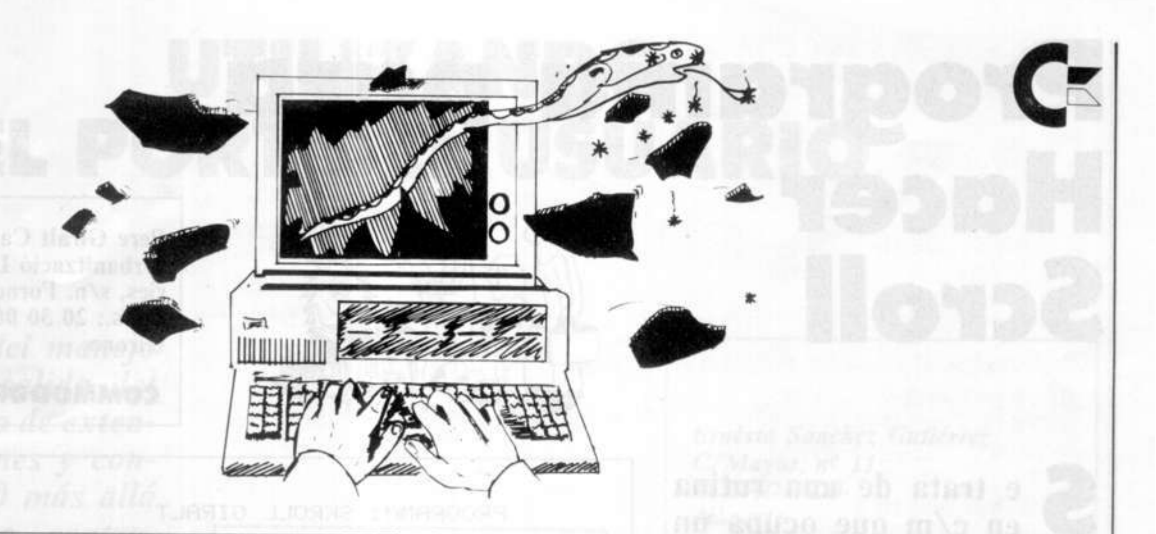
Tu misión es controlar a "Carotena" con la utilización de tu Joystick"

¡Ojo! al cargar el programa, no dejar teclas pulsadas en el "cassette" ya que podría impedir la utilización del Joystick.



**EXPLICACION DEL PROGRAMA**

| | |
|---|---|
| 10-60 | Iniciativas variables. Dimensiones. Colores al borde y de la pantalla "gosub" para presentación e instrucciones de juego. |
| 70-100 | Dibujan el borde de la pantalla (muros eléctricos). |
| 110-160 | Funciones Random para el asterisco y "Carotena". |
| 170-200 | Instrucciones para la utilización de Joystick. |
| 210 | Choque de "Carotena" contra el muro. |
| 240 | Choque de "Carotena" con un asterisco. |
| 250 | Lugar del nuevo asterisco. |
| 260-320 | Localización de la cola de "Carotena". |
| 330 | Sonido de "Carotena". |
| 350 | Movimiento de "Carotena". |
| 360 | Control de velocidad de "Carotena". |
| 370-420 | Presentación del juego. |
| 430-560 | Instrucciones y niveles de juego. |
| 570 | Opción laberinto difícil. |
| 590 | Opción laberinto fácil. |
| 600-690 | Fin de juego. Señalización de puntuación. Comienzo de un nuevo juego. |
| 700-720 | Encuentra la localización para el nuevo asterisco. |
| 730-750 | Rutina Lenguaje Máquina. |
| 760-780 | Laberinto mediante instrucciones Data. |



```
PROGRAMA: CAROTENA

10 DT=60:DIMMA(DT):DIMQ(100)
20 FORJ=0TO65:READJM:POKE828+J,JM:NEXT:F
ORJ=1TODT:READMA(J):NEXT
30 PRINT"[1WHT][1CLR]":POKE36879,111:POK
E36878,15:S3=36877:C=30720:SC=7680
40 MZ=0:P=0:DR=0
50 V=36878:S1=36875:S2=36876:A=2:N=2:MM=
0
60 GOSUB370
70 FORJ=7680TO7700:POKEJ+C,0:POKEJ,160:N
EXT
80 FORJ=7701TO8185STEP22:POKEJ+C,0:POKEJ
,160:NEXT
90 FORJ=8184TO8164STEP-1:POKEJ+C,0:POKEJ
,160:NEXT
100 FORJ=8142TO7702STEP-22:POKEJ+C,0:POK
EJ,160:NEXT
110 M=INT(RND(1)*506)+SC
120 IFPEEK(M)<>32THEN110
130 POKEM,42
140 S=INT(RND(1)*506)+SC
150 IFPEEK(S)<>32THEN140
160 POKES,90
170 SYS828
180 IFPEEK(1)-PEEK(2)=0THEN210
190 DR=PEEK(1)-PEEK(2)
200 IFDR=-21THENDR=1
210 IFPEEK(S+DR)=160ORPEEK(S+DR)=43THENP
OKES,43:POKES+DR+C,2:POKES+DR,90:GOTO600
220 IFMM=1THENGOSUB700
230 SYS828
240 IFPEEK(S+DR)=42THENPOKES1,250:POKES2
,250:SYS828:P=P+1:N=N+2:MM=1:POKES1,0:PO
KES2,0
250 IFMM=0THENPOKEM,42
260 Q(A)=S+DR
270 SYS828
280 Z=A-N
290 IFZ<0THENZ=101+(A-N)
300 POKEQ(Z),32
310 A=A+1:SYS828
320 IFA>100THENA=0
330 POKES2,230:FORT=1TO2:NEXT:POKES2,0
340 SYS828
350 POKES,43:POKES+DR,90:S=S+DR:SYS828
360 FORT=1TOSK:NEXT:GOTO170
370 IFTR=1THENPRINT"[1CLR]":GOTO450
380 N$="[24SPC][1SHIFZ]+++CAROTENA+++++[
3SPC]"
390 FORJ=1TO45:POKES2,230:FORT=1TO2:NEXT
:POKES2,0
400 PRINT"[1HOME][4CRSRD]"MID$(N$,J,22)
410 FORT=1TO150:NEXT:NEXT
420 PRINT
430 FORT=1TO2000:NEXT:PRINT"[1CLR][4CRSR
D]PULSA[1SPC]TECLA[1SPC]STOP/EJECT[1SPC]
"
440 PRINT"[2CRSRD]CONECTE[1SPC]SU[1SPC]J
OYSTICK"
450 PRINT"[3CRSRD]ELEGIR[1SPC]NIVEL[1SPC
](1-9)[3SPC][1CRSRD][1SPC]1)LENTO[1SPC]9
)RAPIDO"
460 GETA$:IFA$=""THEN460
470 IFA$<"1"ORA$>"9"THEN460
480 SK=(10-(VAL(A$)))↑2
490 IFTR=1THENPRINT"[1CLR]":GOTO520
500 PRINT"[1CLR][1CRSRD]DEBERAS[1SPC]COM
ER[1SPC]EL[1SPC]MAYOR[1CRSRD][1SPC]NUMER
O[1SPC]DE[1SPC]ASTERISCOS[2SPC][1CRSRD][
1SPC]QUE[1SPC]PUEDAS."
510 PRINT"[2CRSRD]TU[1SPC]PUNTUACION[1SP
C]ESTA[1SPC]EN[1SPC][1CRSRD][1SPC]RELACI
ON[1SPC]CON[1SPC]EL[1SPC]NIVEL[1CRSRD][1
SPC]QUE[1SPC]ESCOJAS.[1SPC]"
520 PRINT"[2CRSRD]'D'[1SPC]DIFICIL[1SPC]
LABERINTO":TR=1
530 PRINT"'F'[1SPC]FACIL[1SPC]LABERINTO"

540 PRINT"'N'[1SPC]NO[1SPC]LABERINTO"
550 GETB$:IFB$=""THEN550
560 IFB$="N"THENPRINT"[1CLR]":RETURN
570 IFB$="D"THEN PRINT"[1CLR]":FORJ=1TOD
T:POKESC+MA(J)+C,0:POKESC+MA(J),160:NEXT
:MZ=1:RETURN
580 IFB$<>"F"THEN550
590 PRINT"[1CLR]":MZ=2:FORJ=1TO32:POKESC
+MA(J)+C,0:POKESC+MA(J),160:NEXT:RETURN
600 POKES3,230:FORJ=15TO0STEP-.05:POKEV,
J:NEXT:POKES3,0
610 FORT=1TO500:NEXT
620 IFMZ=1THENP=P*5
630 IFMZ=2THENP=P*2
640 R=P*(VAL(A$))
650 PRINT"[1CLR][2CRSRD][1YEL][1SPC]TU[1
SPC]PUNTUACION:"R
660 IFR>HSTHENHS=R
670 PRINT"[2CRSRD][1SPC]RECORD:"HS
680 FORT=1TO3000:NEXT
690 GOTO30
700 M=INT(RND(1)*506)+SC:MM=0:SYS828
710 IFPEEK(M)<>32THENMM=1
720 RETURN
730 DATA169,128,141,19,145,169,0,133,1,1
33,2,169,127,141,34,145,162,119,236,32,1
45
740 DATA208,4,169,1,133,1,169,255,141,34
,145,162,118,236,17,145,208,4,169,22,133
,1
750 DATA162,110,236,17,145,208,4,169,1,1
33,2,162,122,236,17,145,208,4,169,22,133
,2,96
760 DATA142,143,183,184,185,188,189,190,
205,212,222,223,224,225,226,227,234,235,
236,237
770 DATA238,239,249,256,271,272,273,276,
277,278,318,319,141,144,177,178,179,180,
181,192
780 DATA193,194,195,196,229,230,231,232,
265,266,267,268,269,280,281,282,283,284,
317,320
```

# Programa para Hacer Scroll






**Pere Giralt Carreta**
**Urbanització La Tribana, c/Guilleries, s/n. Fornells de la Selva.**
**Telfs.: 20 30 00 - 47 61 87**
**Girona.**


**COMMODORE 64**

**Se trata de una rutina en c/m que ocupa un total de 47 bytes y que puede ser colocada en cualquier parte de la memoria.**

Su función es hacer skroll en pantalla arriba o abajo y en una zona previamente elegida por el usuario, siendo muy útil para presentación de listados, manteniendo siempre un mensaje en pantalla, por ejemplo la cabecera del listado.

Los parámetros de dicha rutina están ubicados en página cero en dos bytes libres para el usuario y son los siguientes:

$FB=251 Primera línea de skroll
$FC=252 Ultima línea de skroll.

Siendo M el inicio de carga de la rutina en memoria: SYS (M) efectuará un skroll arriba y SYS (M+25) lo hará abajo.

En el programa adjunto se ha elegido para M el valor 49152 ($0000) que es un lugar que está a salvo de cualquier invasión del basic.

En dicho programa y para completarlo, después de la carga en memoria de la rutina he puesto un ejemplo en el que se hacen "skrollar" los mensajes de error de C-64 arriba o abajo, según se pulse la tecla de cursor correspondiente.

Esperando que esta colaboración os pueda ser de utilidad me despido de vosotros deseándoos grandes éxitos en la labor que estáis ejecutando.

P.D.: Una aclaración, después de efectuar la rutina de skroll sea arriba o sea abajo el cursor queda siempre a punto para realizar el PRINT o el INPUT en la línea que ha dejado libre el skroll.

La última línea (parámetro $FC3252) no puede tener valor superior a 23, para evitar que el ordenador haga otro skroll por su cuenta (esto sólo sucede cuando el skroll es hacia arriba). ■

```
PROGRAMA: SKROLL GIRALT


1 REM              *SKROLL1*
2 REM             "[9COMMU]"   (P.GIRALT)
3 REM
4 M=49152:REM UBICACIO RUTINA ($C000)
5 REM
6 REM           *CARGA RUTINA*
7 REM
8 FORI=MTOM+46:READBY:POKEI,BY:NEXT
9 REM
10 REM           ----------
90 DATA166,251,32,240,233,189,241,236,13
3,172,181,218,32,200,233,232,228
91 DATA252,208,238,134,214,76,255,233,16
6,252,32,240,233,189,239,236,133
92 DATA172,181,216,32,200,233,202,228,25
1,208,238,240,229
93 REM         **************
94 REM
95 REM
100 REM
105 REM    *PROGRAMA EXEMPLE*
110 REM    ------------------
115 PRINT"[1CLR][4CRSRD]PAUSE[1SPC]PER[1
SPC]CARGA[1SPC]DELS[1SPC]MENSATJES[2SPC]
D'ERROR[1CRSRD][1SPC]EN[1SPC]M$(*)
120 REM
125 L=26:DIMM$(L):B=128:P=41372:FORI=0TO
L:X$=""
130 P=P+1:K=PEEK(P):IFK<BTHENX$=X$+CHR$(
K):GOTO130
135 M$(I)=X$+CHR$(K-B):NEXT
140 REM
145 REM           *SKROLL*
150 REM
155 L1=5:L2=20:REM ZONA DE SKROLL
160 POKE251,L1:POKE252,L2:REM PARAMET.
165 R$="[7SPC][1RVSON][23SPC][1RVSOF]"
170 PRINT"[1CLR]MENSATJES[1SPC]D'ERROR",
,,"[1CRSRD](PULSAR[1SPC]CRSRD[1SPC]O[1SP
C]CRSRU)
175 PRINT"[1CRSRD]"R$"[16CRSRD]",R$
180 UP=M:    REM SKROLL AMUNT
185 DW=M+25:REM SKROLL AVALL
190 GETS$:IFS$=""THEN190
195 IFASC(S$)=145THENSYSUP:GOTO210
200 IFASC(S$)=17 THENSYSDW:GOTO210
205 PRINT"[1CLR]":LIST-95:REM FINAL
210 PRINT"[8CRSRR]"M$(N):N=N+1:IFN>LTHEN
N=0
215 GOTO190
220 END
```



# UTILIZANDO EL PORT DEL USUARIO

**_Trata acerca del manejo del Port de salida del usuario con objeto de extender las aplicaciones y controles del VIC-20 más allá del propio equipo, controlando y recibiendo señales del exterior._**

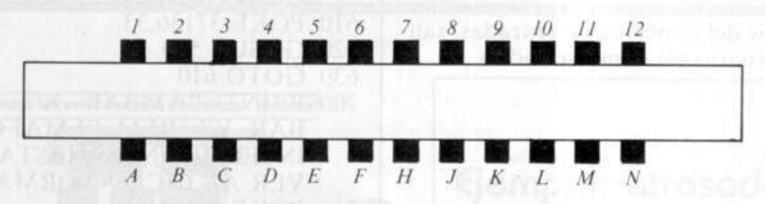

*Ernesto Sánchez Gutiérrez*
*C/Mayor, nº 11*
*ALCADOZO*
*Albacete*




El tema no está desarrollado exhaustivamente, más bien intenta iniciar al usuario, para que con la imaginación que normalmente se posee, pueda expansionarse en este campo.

El VIC-20 dispone de un Port libre para el usuario y accesible directamente por el conector trasero de entradas/salidas usuario. Este Port (que es el Port B del circuito integrado interface 6522) va contraseñado como PB0... a ...PB7, según el esquema del conector, figura 1.

*ENTRADA/SALIDA USUARIO*




| Contacto nº | TIPO | NOTA | Contacto nº | TIPO | NOTA |
|---|---|---|---|---|---|
| 1 | Tierra | | A | TIERRA | |
| 2 | + 5V | 100mA.MAX | B | CB1 | |
| 3 | RESET | | C | PB0 | |
| 4 | Joy 0 | | D | PB1 | |
| 5 | Joy 1 | | E | PB2 | |
| 6 | Joy 2 | | F | PB3 | |
| 7 | Lápiz-óptico | | H | PB4 | |
| 8 | Interruptor cassette | | J | PB5 | |
| 9 | Entrada Serial ATN | | K | PB6 | |
| 10 | + 9V | 100mA.MAX | L | PB7 | |
| 11 | Tierra | | M | CB2 | |
| 12 | Tierra | | N | TIERRA | |

Figura 1

En primer lugar lo que hay que hacer es preparar el Port B para salida de datos, lo que se consigue mediante la instrucción

POKE 37138,A

donde A es un número que puede variar entre 1 y 255 según las salidas (PB0 a PB7) que se pretenda utilizar. Por ejemplo: Si se desea sacar señales por el PB0, PB3 y PB6, el valor de A debería ser

1 + 8 + 64 = 73 = A

| PB7 | PB6 | PB5 | PB4 | PB3 | PB2 | PB1 | PB0 |
|---|---|---|---|---|---|---|---|
| $2^7$ | $2^6$ | $2^5$ | $2^4$ | $2^3$ | $2^2$ | $2^1$ | $2^0$ |

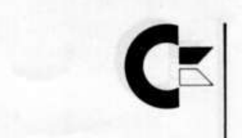
entonces al hacer POKE 37138,73 habremos "abierto" las salidas PB0, PB3 y PB7. Estas salidas estarán ahora a nivel bajo (0 voltios). Si deseamos obtener por ellas un nivel alto de tensión debemos hacer POKE 37136,B

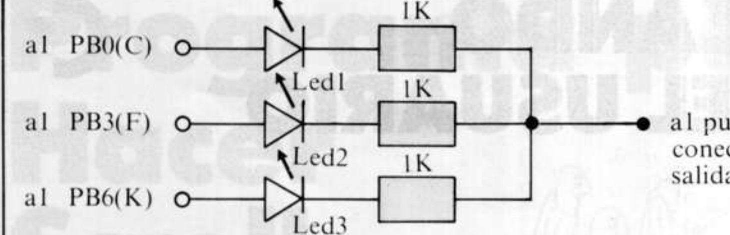
siendo B el número similar al anterior A de tal forma que si B = 73 al hacer POKE 37136,73 las salidas PB0, PB3 y PB7 pasarán al nivel alto de tensión.

Si hacemos POKE 37136,1 sólo pasará la PB0 a nivel alto. Igualmente si ejecutamos POKE 37136,8 sólo pasará la PB3 a nivel alto y si hacemos POKE 37136,64 sólo lo hará la PB6.

Un ejemplo práctico que aclarará aún más las cosas es el que figura al principio de la siguiente página.

Cablear el siguiente montaje






Si escribimos el siguiente programa...

```
10 PRINT "(clr)"
20 PRINT "EMPEZAMOS"
30 POKE 37138,73: REM ABRIR
   EL PORT B DEL USUARIO
   PARA SALIDA
40 POKE 37136,1: REM ENCEN-
   DER EL LED 1 (PB0)
50 GOSUB 500/
60 POKE 37136,8: REM ENCEN-
   DER EL LED 2 (PB3)
70 GOSUB 500
80 POKE 37136,64: REM ENCEN-
   DER EL LED 3 (PB6)
90 GOSUB 500
100 POKE 37136,73: REM ENCEN-
   DER TODOS LOS LED
110 GOSUB 500
120 GOTO 40
500 REM TIEMPO DE ENCENDIDO
510 FOR T = 1 TO 500
520 NEXT T
530 RETURN
```

y lo hacemos rodar (RUN (RETURN)), conseguiremos encender paulatinamente los tres Leds, de uno en uno y al final del ciclo los tres al mismo tiempo. Está claro? Pues imaginad las posibilidades que tenéis.

Os voy a comentar otro ejemplo más práctico del manejo del Port del usuario; controlaremos los semáforos de un supuesto cruce de calles como las representadas en la figura 2.

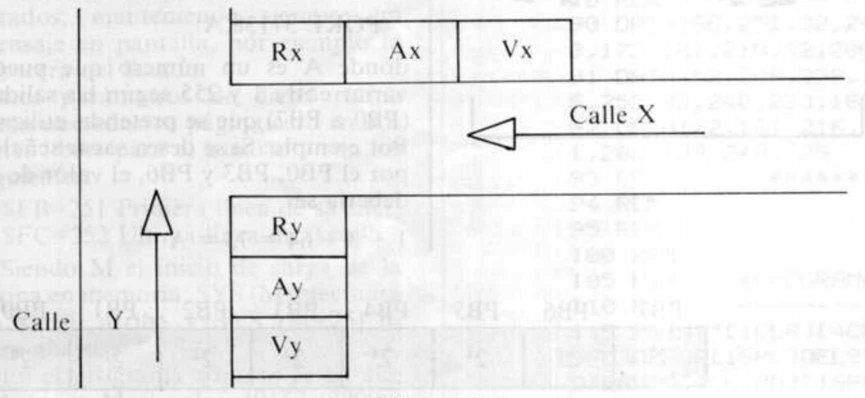



Figura 2

Si realizamos el pequeño montaje de la figura 3 y lo conectamos a los puntos del conector de entradas/salidas usuario tal como se indica

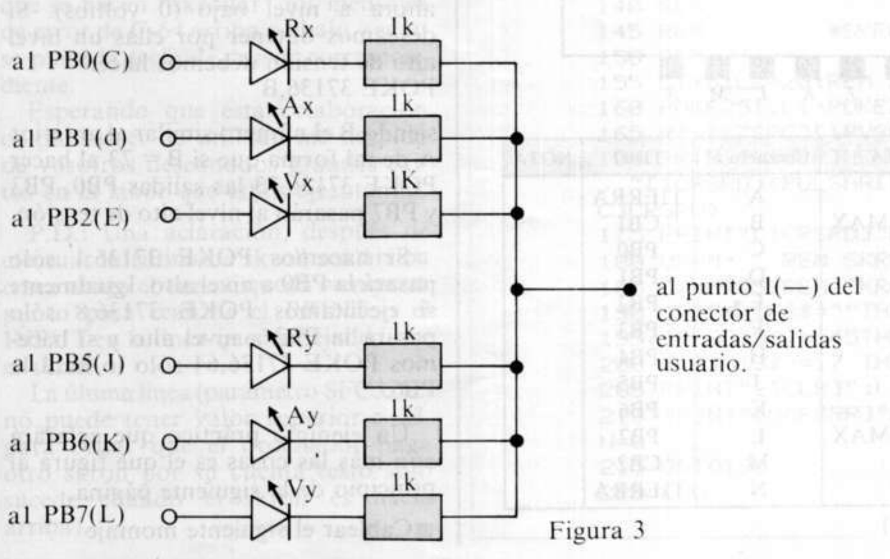



Figura 3

...y escribimos el siguiente programa de control...

```
10 PRINT "(CLR) EMPEZAMOS"
20 POKE 37138,231: REM ABRIR
   EL PORT B PARA SALIDA
   DE DATOS
100 PRINT "ROJO X", "VERDE Y"
110 FOR N=1 TO 500
120 POKE 37136,129
130 NEXT N
200 PRINT "ROJO X", "AMBAR Y"
210 FOR N=1 TO 50
220 POKE 37136,65
230 NEXT N
300 PRINT "VERDE X", "ROJO Y"
310 FOR N=1 TO 500
320 POKE 37136,36
330 NEXT N
400 PRINT "AMBAR X", "ROJO Y"
410 FOR N=1 TO 50
420 POKE 37136,34
430 NEXT N
480 REM VUELTA AL COMIENZO
   DEL CICLO
490 GOTO 100
```

Si pensamos que en el transcurso de cada ciclo se deben producir interrupciones inesperadas, podemos añadir alguna subrutina que por ejemplo ponga en rojo o en ámbar intermitente los semáforos. Estas subrutinas podrían ser así:

```
500 REM INTERRUPCION DEL
   CICLO NORMAL
510 GET A$
520 IF A$="1" THEN 600
530 IF A$="0" THEN 700
540 IF A$"£" THEN 550
550 RETURN
600 PRINT "ROJO X", "ROJO Y":
   REM SEMAFOROS A ROJO
   HASTA VOLVER A SITUA-
   CION NORMAL
610 POKE 37136,33
620 GOSUB 500
630 GOTO 610
700 PRINT "AMBAR X", "AM-
   BAR Y": REM SEMAFOROS
   INTERMITENTES HASTA VOL-
   VER AL CICLO NORMAL
710 POKE 37136,66
720 FOR T=1 TO 500
730 NEXT T
740 GOSUB 500
750 POKE 37136,0
760 FOR T=1 TO 500/
770 NEXT T
780 GOSUB 500
790 GOTO 710
```

Habrá que incluir en el programa principal las llamadas a las subrutinas de interrupción, escribiendo...

```
115 GOSUB 500
215 GOSUB 500
315 GOSUB 500
415 GOSUB 500
```



Así si durante un ciclo normal se desea una interrupción, se podrá lograr pulsando "1" o "0" y los semáforos se pondrán en rojo o en ámbar intermitente respectivamente, no saliendo de esa situación hasta que se pulse la tecla "£" retornando entonces al ciclo normal.

Igual que este ejemplo se podía haber programado una secuencia de luces, etc...

Evidentemente en caso de querer controlar algo de más potencia que unos simples Leds, deberemos colocar un interface adecuado a la potencia controlada, de tal forma que no se dañe el Port B utilizado del VIC-20.

Hasta aquí hemos utilizado las posibilidades del Port, para enviar señales al exterior, bien por programa fijo o por interruptores controladas por el teclado. Pero podríamos pensar en alguna aplicación en la que la señal de mando provenga del exterior del equipo, como si fuera un termostato, presostato, final de carrera, etc...

En ese caso podemos utilizar el Port A del mismo circuito integrado interface 6522. O sea lo que conocemos como JOY0, JOY1, JOY2 y LITE PEN, que son los mas directamente accesibles.

El POKE que vamos a utilizar para controlar si alguna de estas entradas ha pasado a 0 voltios es el 37137, de tal manera que podemos efectuar las condiciones...

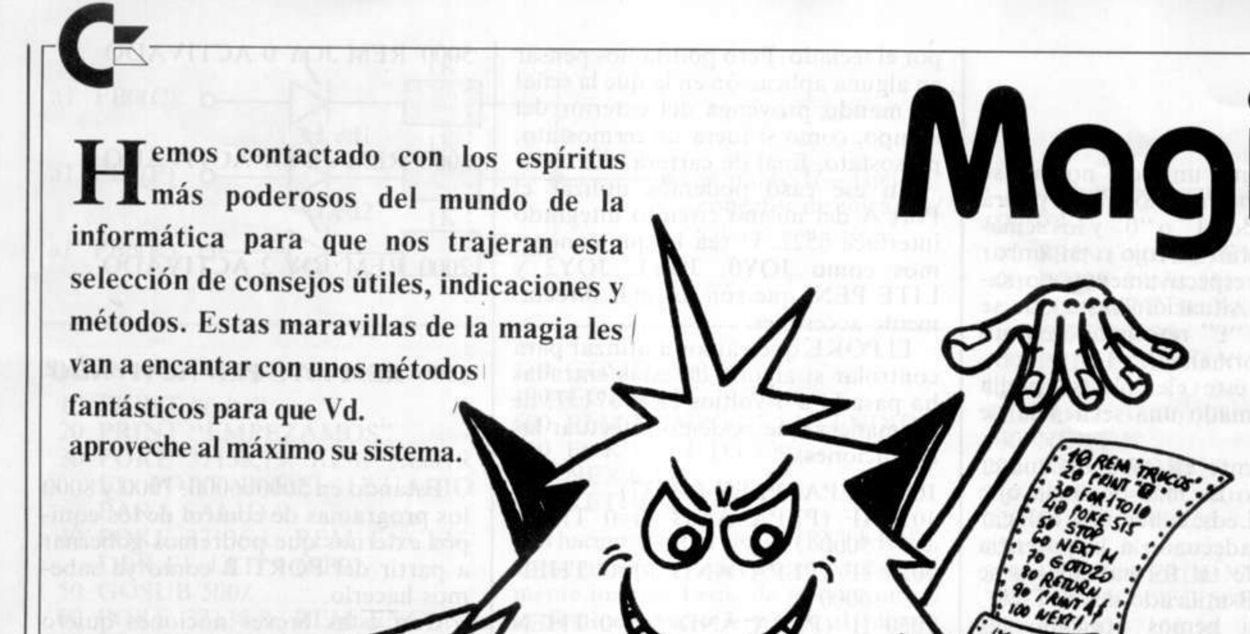
```
1000 PEPA=PEEK (37137)
1010 IF (PEPA AND 4)=0 THEN
     5000
1020 IF (PEPA AND 8)=0 THEN
     6000
1030 IF (PEPA AND 16)=0 THEN
     7000
1040 IF (PEPA AND 32) =0 THEN
     8000
5000 REM JOY 0 ACTIVADO
:
:
6000 REM JOY 1 ACTIVADO
:
:
7000 REM JOY 2 ACTIVADO
:
:
8000 REM LITE PEN ACTIVADO
:
:
```

Estando en 5000, 6000, 7000 y 8000 los programas de control de los equipos externos que podremos gobernar a partir del PORT B como ya sabemos hacerlo.

Con estas breves nociones quiero abrir la puerta para que podáis ampliar las posibilidades de vuestro VIC-20.

# Magia

**Hemos contactado con los espíritus más poderosos del mundo de la informática para que nos trajeran esta selección de consejos útiles, indicaciones y métodos. Estas maravillas de la magia les van a encantar con unos métodos fantásticos para que Vd. aproveche al máximo su sistema.**




*La MAGIA son trucos, la MAGIA es divertida,*
*La MAGIA es hacer lo que nadie se ha atrevido.*

*La MAGIA es una columna mensual llena de consejos, trucos, de esto y aquello del mundo del software, hardware y aplicaciones.*

*Cada mes, MAGIA les trae trucos breves y útiles de informática procedentes de todo el mundo - trucos descubiertos por los demás que hacen que la informática sea más fácil, más divertida o más animada.*

*MAGIA habla de ideas sencillas, programas de una sola línea, subrutinas útiles, hechos de informática poco conocidos y otras cosas de interés. Buscamos material nuevo o renovado que resulta ser de valor actual para usuarios de equipos Commodore y que puede utilizarse con un mínimo de tiempo, esfuerzo o conocimientos teóricos.*

*MAGIA resulta ser la fuente más completa de información para la informática práctica, además de ser un foro internacional para compatir trucos con otros aficionados. Envíe sus trucos a:*

*COMMODORE WORLD*
*Pedro Muguruza, 4*
*Madrid-16*

*La revista "Commodore World" sorteará seis paquetes de software en julio y diciembre entre todas las contribuciones publicadas.*

El truco especial "de una sola línea" de este mes es bastante antiguo —es de 1978—; cuando un Commodore PET de 8K no valía lo que vale ahora y era imposible conseguir ningún tipo de documentación para leer. No había libros, y las únicas revistas eran hojas informativas producidas por usuarios no profesionales.

The PET Gazette era una de estas hojas, y aquí ofrecemos uno de sus trucos más antiguos, llamado "BURROW":

```
1A$="arriba)(abajo)(izquierda)(derecha): PRINTMID$
(A$,RND(.5)*4+1,1)"*(izquierda)";:FORI=1T030:NEXT:
PRINT"(rvs on)(espacio)(izquierda)";:GOTO1
```

Cabe en una línea de 40 columnas, y se hace cada vez más divertido.

Nos gustaría ver sus programas de una sola línea, y queremos publicar por lo menos uno cada mes. Los programas pueden ser divertidos, de humor, útiles o no, con tal de que quepan en 40 columnas o menos. ¿Y usted, qué nos puede ofrecer?

**Montse Ferrán**

* * *

Muchas sentencias caben en una línea de programa mediante el uso de las abreviaturas de las palabras clave de Basic incluidas en el apéndice del manual del usuario. Cuando la línea se produce en forma de listado, las palabras clave se imprimen sin abreviar, con el resultado de que la línea del programa podría ocupar más que el número normal de líneas en la pantalla.

Esto no es problema, pero al intentar editar esta línea larga, el ordenador la reducirá para que se ajuste a la longitud normal para una línea de programa. Así que se pueden utilizar las abreviaturas para poder incluir las sentencias en una línea, pero tenga mucho cuidado al editarla más adelante.

**Pittsburgh Commodore Group Newsletter**

* * *

Todos los 16 colores del Commodore 64 pueden ser llamados desde el teclado, pero sólo ocho de ellos pueden ser reconocidos en las teclas. Si vd. pega un trozo de cinta adhesiva de unas 6 pulgadas encima de las teclas numéricas de 1-8, puede marcar los otros colores en dicha cinta y todo el proceso resulta mucho más fácil.

Los colores se llaman al pulsar la tecla Logo del Commodore junto con una tecla numérica. Empezando por la izquierda los colores son, naranja, marrón, rojo claro, gris oscuro, gris mediano, verde claro, azul claro y gris claro.

**L.F.S.**

* * *

Puede ocurrir que en algunas televisiones la pantalla del VIC no quede exactamente en el centro, con el resultado de que se pierde parte de la representación. La posición 36864 controla el centro horizontal de la pantalla (normalmente 5), y la posición 36864 controla el centro vertical de la pantalla (normalmente 25). Si vd. cambia estos valores le ayudará a centrar la pantalla de una forma más adecuada.

**Westmoreland Commodore Newsletter**

La llamada "modalidad de comillas" puede volverle loco cuando el ordenador se encuentra en dicha modalidad y vd. quiere salir de ella. Normalmente se sale de la modalidad de comillas al teclear más comillas, y luego borrarlas.

Pero hay ocasiones en que esto no funciona, como cuando están llenándose los espacios abiertos mediante la tecla de inserción. Dichos espacios se comportan como si la modalidad de comillas fuese activa, aunque no lo sea, y si se teclea otra comilla no se soluciona nada. Si vd. pulsa cualquier tecla de control del cursor cuando se encuentra con un espacio de inserción, casi siempre se consigue la versión de la modalidad de comillas de dicho control de cursor, que normalmente no es lo que estamos buscando. Si intenta borrarlo, se consigue la T del campo invertido, que obviamente, complica aún más el problema.

¿Cuál es la solución? Pulse la tecla de "shift return", que desplaza el cursor a la línea siguiente sin "introducir" la línea que se está modificando. También borra la modalidad de comillas en los espacios de inserción de forma que el cursor pueda volver a dicha línea para seguir con su trabajo.

**Tomás Jáuregui**

* * *

La expresión matemática entre If y Then determina si el resto de una sentencia será ejecutada. Cuando la expresión es falsa, el resto de la línea se salta.

Esta característica puede utilizarse para ahorrar el tiempo de ejecución. En vez de usar una sentencia como 100 IF X= 1 AND Y= THEN PRINT Z, resulta mucho más rápido escribir 100 IF X = 1 THEN IF Y = 2 THEN PRINT Z.

En el primer caso, X = 1 AND Y = 2 tiene que ser evaluado antes de tomar cualquier decisión sobre el salto de línea. En el segundo caso, en cuanto se evalúe X = 1 como falso, todo lo demás se salta. Esto resulta en una ejecución más rápida cuando X = 1 es falso.

**Roberto F. Rodríguez**

* * *

Aquí tiene la forma de cronometrar la ejecución de dos programas semejantes:

```
100 TI$ = "000000"
110 FOR1 = 1 TO 500
120 Se incluye aquí el código que se comprueba
130 Etc.
140 Etc.
180 NEXT
190 PRINT TI
```

Se procesa el programa con una versión del programa, y se anota el valor del TI, que resulta ser el número de segundos que tardó en ejecutarse 500 veces. Luego se sustituye el programa de prueba con la otra versión y se vuelve a procesar el programa. La versión que requiere menos segundos en ejecutarse es la más rápida.

**L.F.S.**

* * *



Con frecuencia se consigue un número negativo al comprobar FRE(0) en el Commodore 64. (Esto no significa que la memoria disponible es negativa; tiene que ver con la forma en que FRE representa los números). Para convertir el valor negativo en su forma correcta, se realiza lo siguiente:
PRINT FRE(0) + (up arrow) 16

Además, el cero en FRE(0) no es nada mágico; cualquier letra o número puede incluirse aquí. Normalmente resulta más fácil encontrar FRE(9) en el teclado, y da el mismo resultado.

**Carmen Echevarría**

* * *

Utilice tiras de Scotch Magic Tape (papel de celo) para etiquetar las cassettes; se puede escribir claramente con lápiz y borrarlo fácilmente. Pero tenga cuidado de que los trozos de la goma de borrar no caigan dentro de la cinta o la cassette.

**Jordi García**

* * *

Al guardar un programa en una cinta, añada el RVS al nombre del programa. De esta forma, al volver a leerlo, imprimirá el nombre sobre un fondo blanco, facilitando su localización. Simplemente teclee SAVE "(rvs on) NOMBRE DEL PROGRAMA (rvs off)" (return).

**The PET Gazette**

* * *

Si vd. tiene un programa almacenado en la memoria y quiere ejecutarlo, no hace falta teclear la palabra RUN. Simplemente teclee cualquier letra o letras (números no), y pulse la tecla en "SHIFT" run/stop. El programa se ejecutará.

**Moncho Zabaleta**

Si vd. ha quitado el precinto de una cinta que la protege para que no pueda grabarse encima, existen dos formas de burlarse de esta protección. Una forma es tapando el agujero con un trozo de papel de celo. Otra es engañar el Datacassette para que piense que el agujero se ha tapado.

Abra la funda del Datacassette y localice el alfiler pequeño que se encuentra en el agujero vacío. (Se encuentra muy al fondo, en la extrema izquierda). Empuje suavemente el alfiler hacia la parte de atrás del aparato y pulse el botón de grabar. Introduzca la cinta y vuelva a pulsar el botón de grabar mientras pulsa el botón de play.

**Frank O'Conner**

* * *

No hace falta colocar paréntesis alrededor del número después de una sentencia SYS. SYS828, SYS 828 y SYS(828) significan lo mismo para el ordenador.

**L.F.S.**

* * *

Podría resultar problemático introducir gráficos o letras en "SHIFT" en una sentencia REM - salen en el listado de una forma rara, normalmente como palabras clave de Basic. Para que se comporten como es debido, póngalos entre paréntesis.

**J. Luis G. Diego**

* * *

Una forma rápida de saber si un número entero es par o impar es hacerle un AND con un 1. Si el resultado es cero, el entero es par; si el resultado es uno, el número es impar. Para que funcione el truco, el número tiene que encontrarse dentro de -32768 a +32767; si no es así, el resultado será un error de cantidades no válidas.

**Iñigo Fernández**

# MARKETCLUB

**—Servicio gratuito para nuestros lectores particulares. Empresas 300 ptas. por línea.**

- **CBM 4.032. Intercambio programas. José Marcé Mestres. Cálle Sevilla, 5. Tel. (93) 803 77 51, de 8 a 3. VILANOVA DEL CAMI (Barcelona).**
- **ACCESORIOS VIC-20: Ampliación de memoria 16K más varios programas, 13.000. Módulo de expansión para 6 cartuchos, 10.000. Cartucho lenguaje FORTH y manual, 7.000. Las tres cosas sólo por 25.000. Jaime. Tel. 245 46 56. BARCELONA.**
- **En Barcelona, clases de informática. PLAZAS LIMITADAS. Lenguaje BASIC. Prácticas con micro-ordenador VIC-20. Prof. E. Martínez de Carvajal. Información: Tel. (93) 345 10 00. Señorita María José (mañanas) o (93) 345 87 75. Sr. Martínez (fuera de horas de oficina).**
- **Desearía vender por 40.000 ptas.: VIC-20, con cartucho Super Expander, las dos partes del Curso, el "joy-stick" y un juego Indescomp, todo comprado en diciembre del 82. Fernando Martínez, calle La Roda, 39, 5º D. Teléfono 23 41 82. ALBACETE.**
- **Tengo un PET 2001/8K y desearía tener el cassette "Monitor para lenguaje Máquina". También desearía contactar con usuarios o clubs de PET si los hay. José Manuel Cámara Mas, calle Castor, 32, bloque II, 3º, puerta 1. ALICANTE.**
- **Programación de ordenadores personales; organización explotación de ficheros; programas ordenadores auxiliares, para cuestiones empresariales, profesionales, administrativas, científicas. Mora Mas. Carlos III, 41. Teléfono 339 98 29. BARCELONA-28.**
- **Vendo impresora Seikosha GP-100-VC para VIC-20, Commodore 64. Como nueva: 43.000 ptas. Regalo muchos programas gráficos al comprador. Llamar horas comida y cena. Fco. Gutiérrez. Tel. 253 13 40. Santiago Rusiñol, 12. MADRID-3.**
- **Vendo VIC-20 con expansión 16K, superexpander, curso programación BASIC (los dos tomos), guía de referencia del programador y cartucho de juegos. Todo en perfecto estado. Precio: 55.000 ptas. Angel Jiménez. Teléfono (954) 458 320. SEVILLA.**
- **Vendo VIC-20, ampl. de memoria 16K RAM, cassette. Libros: Manual VIC-20, Reference's, VIC Revealed y 25 programas muy buenos: "Comecocos", "Blitz", "S. de Ecuaciones", "Matrices", etc. Por sólo 55.000 ptas. David Badalona, 102. Tel. (91) 734 11 03. MADRID-34.**
- **Tengo programa para confección de documentos, cartas, textos, etc. Permite escribir líneas reales de impresora visualizando por pantalla y avisando el final de cada línea por timbre. Visualización del texto entero; modificaciones de línea; insertar líneas; grabación disco o cassette; lectura de datos de disco o cassette; control por pantalla del texto a rectificar; acepta,;: espacios, etc.; imprime normal y doble ancho; pregunta número de copias; salto automático de página en impresora; margen izquierdo en la página. Amadeo Bargay. Plza. Hospital 5-5º. Teléfono 874 41 93. Manresa. BARCELONA.**

# CBM - B 700 (3)

# SOFTWARE PARA EL 700



***Los capítulos previos fueron publicados con anterioridad en COMMODORE CLUB. Aquellos lectores que los deseen, les rogamos nos lo comuniquen para poder enviarles copia de estos capítulos.***

JORDI SASTRE

En los dos artículos anteriores he descrito algunos de los más importantes aspectos del 700 a nivel de características técnicas y sistema operativo. Pero el aspecto más importante en todo ordenador es el software, y el 700 no puede ser una excepción. Cuando una persona (o empresa) decide instalar un ordenador es por alguna de las siguientes causas:

—El trabajo administrativo desborda al departamento de administración.

—Los directivos precisan de una información (estadística), que exige gran trabajo de elaboración manual, mientras que un ordenador puede suministrarla rápida y fácilmente.

—Debido al volumen de trabajo del departamento de contabilidad, almacén, estadística o facturación, continuamente se están corrigiendo errores, pensando en qué pasará con los errores que no pueden corregirse porque no se han detectado.

—Se desea efectuar un control más estricto sobre los clientes de la empresa (riesgo, impagados...) para evitar "sorpresas" financieras.

—Actualmente se están contratando los servicios de un centro de cálculo. Pero el coste del mismo, el inconveniente de no tener la información "a mano", y el peligro que implica el que se manejen los datos de la empresa fuera de ella, hacen pensar que es mejor disponer de un ordenador propio.

—Por causas diversas: porque fulanito tiene uno, porque un amigo me lo aconseja (generalmente este amigo es representante de alguna marca)... etc.

Entonces, cuando la empresa envía a una persona para que "busque ordenador", se encuentra con que no tiene ningún conocimiento de informática y se da cuenta de que fácilmente puede ser objeto de engaños. Una posible solución es contratar los servicios de una empresa asesora en informática para que le mecanice la empresa. Pero actualmente, el mundo de los ordenadores, y más concretamente el mundo de los micro-ordenadores, está creciendo de tal manera, que nadie puede estar totalmente al día en cuanto a máquinas y programas disponibles en cada momento. Cada día aparecen nuevos paquetes standard de contabilidad, facturación o nóminas, siempre mejores que los anteriores. Cada día aparecen en el mercado nuevos ordenadores, en principio mejores que los anteriores (pero no todos). Un día el ordenador vale tanto, pero al día siguiente ha aparecido un modelo mejor o más barato. En resumen, según el día de la semana en que se empiece a "buscar ordenador", se acabará comprando una marca u otra.

Actualmente, las empresas distribuidoras de micro-ordenadores buscan el mayor número de programas standard posible. Porque un cliente puede ver enseguida si un programa le sirve o no, siempre y cuando este programa esté hecho y vea cómo trabaja. Los programas a medida no se ven hasta que están hechos, por lo que el cliente no sabrá si el ordenador cumplirá sus exigencias hasta que el programa esté acabado, después de largas semanas de lucha con el programador, y siempre recordándole las promesas del vendedor.

El programa standard tiene el inconveniente en él mismo. En la mayoría de los casos (sobre todo en gestión comercial y facturación) cuando una empresa adopta un programa standard es para adaptarse a él, en lugar de adaptar el ordenador a la empresa, que sería lo más deseable.

Un punto intermedio entre el programa standard y el programa a medida, es el programa a medida confeccionado muy rápidamente con un buen sistema operativo. ¡Aquí quería llegar!

Más importante que las características externas, más importante que todos los programas llamados standard que pudiera haber, y más importante que cualquiera de las promesas



que pueda hacer un vendedor, el sistema operativo es quien define en verdad la potencia del ordenador. De acuerdo que un buen sistema operativo debe ir soportado por una buena máquina, pero hay que saber por dónde empezar a mirar.

El sistema operativo y, por tanto, el lenguaje o lenguajes de programación que utilice, intervienen en muy alto grado en la velocidad de programación de los ordenadores. Para hacer un programa podemos estar tres días o tres meses, dependiendo del sistema operativo que estemos empleando.

He aquí la mejor política que puede tener una distribuidora de micro-ordenadores: tener como standard los programas que de verdad pueden serlo (contabilidad, proceso de textos, visicalc, etc...), y disponer de un buen sistema operativo para que los programas a medida puedan hacerse en el mínimo de tiempo, para que el cliente final vea rápidamente si el ordenador le sirve o no, y para bajar los costes de programación, que últimamente se están disparando.

Esto es lo que se pretende con la serie 700. Pronto estarán disponibles los programas de proceso de textos y Calc Result. Lo que sí ya está hecho y disponible es MEC/DOS. MEC/DOS es un sistema operativo co-residente con el sistema operativo normal del 700, que amplía en unos 50 comandos el Basic de Commodore. Sus objetivos son simplificar la programación, aumentar la velocidad de ejecución..., en una palabra, la potencia del ordenador. Pero MEC/DOS es tema para otro artículo. En todo esto estaba pensando MEC-SOFT cuando desarrolló MEC/DOS. MEC/DOS permite que los programas sean más fáciles de confeccionar, más rápidos en su ejecución, más presentables en su edición de pantallas, más potentes en cálculo matemático, más claros para posteriores modificaciones más... y más...

En breve, el 700 dispondrá de procesadores de textos (Wordcraft, Easyscript, Superscript), programas tipo visicalc (Calc Result), bases de datos, etcétera.

En el próximo artículo comentaré más profundamente lo que es el MEC/DOS.

# MEA CULPA:

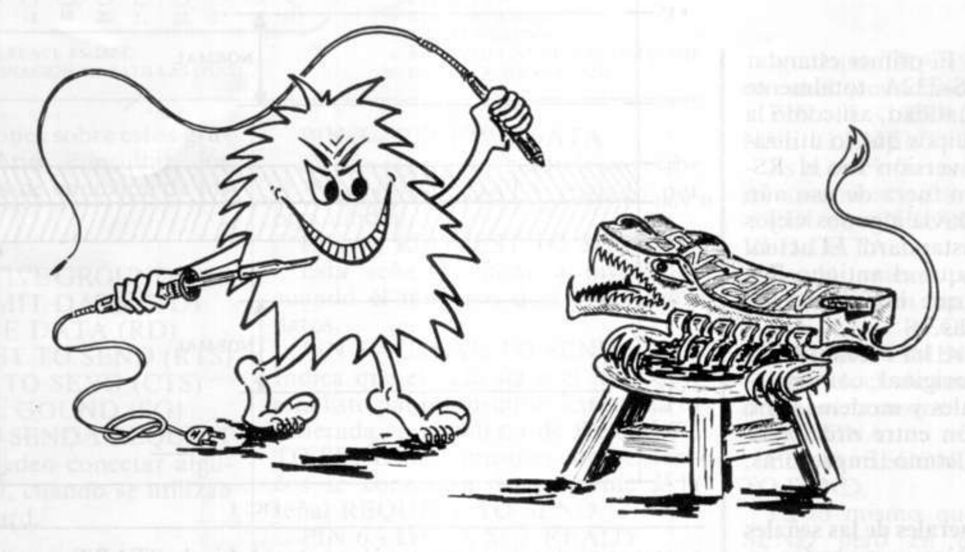
*DEBIDO A PROBLEMAS EN EL MANUAL DEL PROCESO DE TEXTO "EASY SCRIPT" PARA EL COMMODORE 64 NO ESTA SUFICIENTEMENTE CLARO COMO SE CARGA EL PROGRAMA EN SU VERSION ACTUAL EN DISCO. DESPUES DE ENCENDER EL EQUIPO PULSAR:*

*LOAD "*", 8, 1*

*SEGUIDO DE RETURN Y EL PROCESO DE TEXTO EMPEZARA A FUNCIONAR.*

# EL RINCON DEL VICCOLAGE





**Los capítulos previos fueron publicados con anterioridad en COMMODORE CLUB. Aquellos lectores que los deseen, les rogamos que nos lo comuniquen para poder enviarles copia de estos capítulos.**

# COMUNICACION SERIE MEDIANTE EL BUS RS-232C (I)

**A juzgar por las preguntas que recibimos en nuestra redacción, existe una clamorosa demanda de información sobre el manejo del interface serie (RS-232) que equipa a los equipos VIC-20 y COMMODORE 64. Como Manel Sans es quien tiene la mayor experiencia en este tema, le hemos pasado la "pelota". El ha preparado para nuestra revista una interesante serie de artículos que derramaran billones de fotones sobre los rincones más oscuros de este sistema estandar de comunicación.**

M. SANS

En términos generales existen dos métodos básicos para comunicar un ordenador con sus periféricos o con el mundo exterior. Uno es la transferencia de datos en paralelo y el otro es la comunicación serie o transferencia de datos en serie.

La transferencia de datos en paralelo implica enviar o transmitir 8 bits de datos mediante 8 hilos separados y transmitir un byte de información en cada transferencia. De todas formas para efectuar una transmisión fiable de información serán necesarias algunas señales de control adicionales. Estas señales de control se llaman en inglés "HAND-SHAKING", que podríamos traducir por "pautas (o señales) de acuerdo".

Algunos ordenadores utilizan el protocolo de comunicación paralelo IEEE-488 establecido en 1978. Otro standard de manejo de bus muy común es el CENTRONICS PARALLEL INTERFACE STANDARD. Este método es el más comúnmente utilizado en muchas impresoras.

El uso del bus paralelo en largas distancias produce muchos problemas ya que, por cada cable circulan diferentes corrientes a diferentes velocidades. Esto tiene como efecto que en distancias grandes los bits del dato enviado lleguen a su destino a diferente tiempo.

Aun cuando muchos fabricantes utilizan el sistema IEEE-488 de comunicación, el método más común es la comunicación serie según la norma RS-232C. La comunicación por este sistema se efectúa bit a bit, a una cierta velocidad y por un solo hilo.

La mayor ventaja de este sistema es la posibilidad de transmitir los datos a grandes distancias. Con las modificaciones adecuadas este método se puede utilizar para transmitir datos por vía telefónica o por vía satélite.

Esencialmente RS-232C es el nombre para el estandard formulado por la ELECTRONIC INDUSTRIES ASSOCIATON (EIA). Este estándar describe un juego de condiciones necesarias (PROTOCOLO), para el manejo del bus serie entre dos usuarios.

El RS-232 ha existido en tres diferentes formas desde su formulación a



principios de 1960. El primer estandar RS-232 fue el RS-232A totalmente obsoleto en la actualidad, así como la mayoría de los equipos que lo utilizaban. La segunda versión fue el RS-232B, esta también fuera de uso aún cuando existen todavía algunos viejos equipos con este estandard. El actual RS-232C es igual que el antiguo RS-232B excepto en que las señales de datos son invertidas.

Este estandard se ha extendido ya no sólo a su área original, comunicación entre terminales y modems, sino como interconexión entre ordenadores y periféricos como impresoras, plotters, etc.

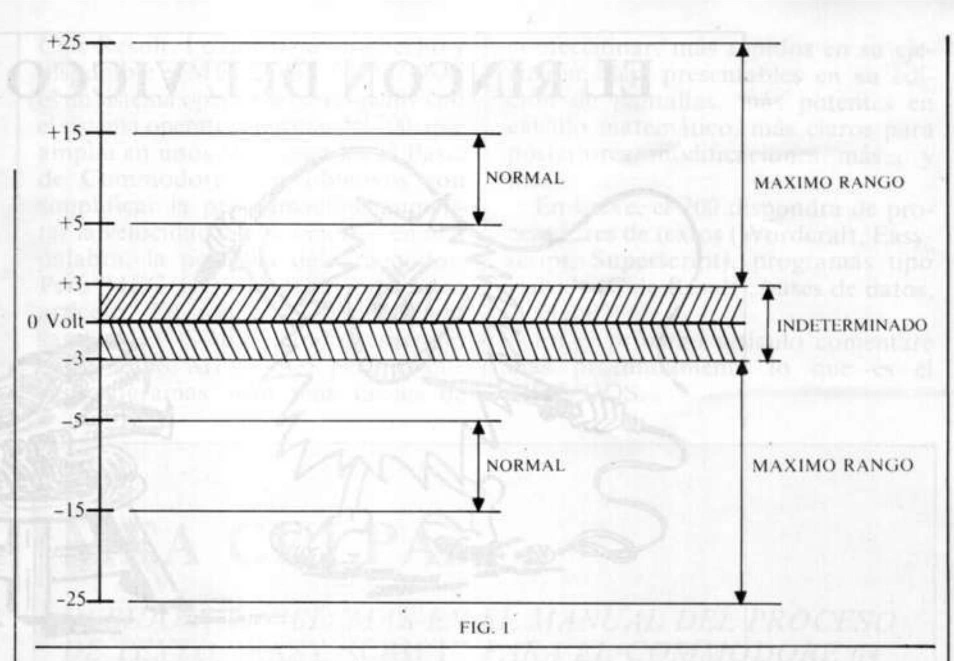


FIG. 1

**Características generales de las señales**

Algunos de los términos usados en comunicación provienen de los ancestrales TELETYPE. Los términos para uno y cero lógico, por ejemplo, son MARK y SPACE respectivamente. La utilización de estos términos en comunicación indicaba la presencia o ausencia de una corriente de 20 miliamperios en el circuito. Algunos teletipos utilizan esta técnica.

El RS-232C ha retenido estos términos, pero usando tensiones para la transmisión de datos en lugar de corrientes. La condición de SPACE ("0") se representa por la presencia de una tensión comprendida entre +3 y +25 voltios y en cambio la condición MARK ("1") se representa por la presencia de una tensión comprendida entre -3 y -25 voltios. Un punto de trabajo entre 5 y 15 voltios es el más usual. Si la tensión positiva o negativa de la transmisión está por debajo de los 3 voltios se produce un estado de indeterminación y el receptor no podrá definir o detectar la condición de esta señal. (Ver figura 1).

Cuando se efectúan transmisiones a través de grandes distancias, la señal en la linea es atenuada ya que como es sabido, el voltaje es inversamente proporcional a la distancia de cable atravesado. El límite práctico en la longitud del cable para este tipo de transmisiones es de alrededor de 300 metros. Para poder transmitir a más largas distancias se debe utilizar buffers de línea especiales que mantengan la señal. Otro método para cubrir largas distancias es el uso de un MODEM (MOdulador/DEModulador).

Otro problema común en la utilización de grandes longitudes de cable son las interferencias. Estas interferencias son en su mayor parte de naturaleza electromagnética. Algunos ejemplos de fuentes de interferencias electromagnéticas son, luces fluorescentes, circulación o colocación de los cables cerca de transmisiores de radio o televisión, proximidad de instalaciones de RADAR, motores eléctricos, transformadores y muchos más. Para evitar o minimizar estos efectos es aconsejable la utilización de cables apantallados y/o con conductores trenzados. En el caso de que alguno de los conductores no sea utilizado es conveniente unirlo a masa.

**Modos de transmisión**

En RS-232C son posibles dos modos de transmisión:

1) TRANSMISION SINCRONA
2) TRANSMISION ASINCRONA

La transmisión SINCRONA mantiene un intervalo de tiempo fijo entre los bits de la transmisión. Los períodos de tiempo son definidos por la estación transmisora mediante el TRANSMIT CLOCK, con el cual la estación receptora sabe exactamente cuando el dato o los bits de dato son válidos.

Los datos SINCRONOS están normalmente organizados en grupos rígidos de caracteres que son llamados colectivamente "PROTOCOLO".

La ventaja en la utilización de la comunicación SINCRONA es que la estación transmisora tiene un control rígido sobre el receptor, y por esta causa el formato del mensaje se conoce por anticipado y los cheksums usados para la detección de errores son muy precisos. La principal desventaja en este tipo de transmisión es lo costoso de mantener todos los tipos de mensajes y respuestas. Por esta limitación es muy raro ver transmisores SINCRONOS en microcomputadores.

La transmisión ASINCRONA es la más común de los dos tipos. Esta no depende del clock hasta después de haber detectado el bit de START. Ese bit de START se utiliza para comunicar que se inicia la transmisión de un carácter y también se puede utilizar para determinar la longitud del tiempo de bit.

El número de bits en el carácter puede variar, el código original de 5 bits o BAUDOT es el código utilizado en teletipos y es el que más facilidades nos ofrece para la conexión con ellos, (internacionalmente el telex utiliza 5 bits). El código BAUDOT de 6 bits es utilizado en algunos casos, pero el código de 7 bits (usualmente ASCII) es el más utilizado comúnmente.

La utilización de un código de 8 bits puede efectuarse de dos maneras completamente diferentes; la primera es utilizar un código de siete bits y el octavo utilizarlo como bit de paridad y la segunda es utilizar un código de 8 bits, por ejemplo el ASCII extendido.

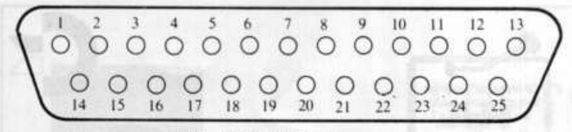
Además en el formato de información enviado existe un bit de paridad que puede ser par, impar o no utilizarse y que no tiene nada que ver con la paridad generada para el octavo bit de dato como comentábamos antes.

El bit que se envía al final de estos caracteres ASINCRONOS es el bit de STOP. La longitud de este bit puede variar entre 1, 1,5 y 2 tiempos de bit. Este bit es siempre un nivel MARK ("1") y finaliza con el bit de START del siguiente carácter. (Ver figura 2).

La mayor ventaja en la comunicación ASYNCRONA es la relativa facilidad de uso y el hecho de no requerir grupos de caracteres. Esto es particularmente conveniente para el uso con microcomputadores donde es esencial el espacio de memoria. La principal desventaja de la comunicación ASINCRONA es que la paridad es la única protección contra cualquier error.

**Señales de control**

Existen 3 grupos de señales de con-




1. PROTECTIVE GROUND
2. TRANSMIT DATA
3. RECEIVE DATA
4. REQUES TO SEND
5. CLEAR TO SEND
6. DATA SET READY
7. SIGNAL GROUND
8. RECEIVED LINE SIGNAL DETECTOR
20. DATA TERMINAL READY

FIG. 3 INTERFACE RS-232-C ASIGNACION DE PATILLAS (PINS)

trol. Para variaciones sobre estos grupos será necesario consultar los manuales técnicos de conexión de los aparatos.

GRUPO 1:
PIN 1 - PROTECTIVE GROUND (PG)
PIN 2 - TRANSMIT DATA (TD)
PIN 3 - RECEIVE DATA (RD)
PIN 4 - REQUEST TO SEND (RTS)
PIN 5 - CLEAR TO SEND (CTS)
PIN 7 - SIGNAL GOUND (SG)
Nota: CLEAR TO SEND Y REQUEST TO SEND se pueden conectar algunas veces entre sí, cuando se utilizan interfaces estandard.

GRUPO 2:
Consiste en el grupo 1 más:
PIN 6 - DATA SET READY (DSR)
PIN 8 - DATA CARRIER DETECT (DCD)
PIN 20 - DATA TERMINAL READY (DTR)

El grupo 2 es usado principalmente en conexiones entre terminales y CPU's o en transmisiones entre CPU y CPU.

GRUPO 3:
Consiste en los otros dos grupos más:
PIN 22 - RING INDICATOR (RI)
Este grupo se encuentra utilizado normalmente en estaciones fijas de MODEM's AUTO-ANSWER.

En la figura 3 tenemos un diagrama que representa un conector RS-232C con asignación de pins.

PIN 1 - PROTECTIVE GROUND
Esta es la masa de chasis o masa de seguridad. Cuando se utiliza cable apantallado la malla se ha de conectar a este pin.

PIN 2 - TRANSMIT DATA
Es por este pin por donde se envían los datos al exterior. Esta señal está a nivel MARK ("1") cuando no se utiliza. Antes de una transmisión, REQUEST TO SEND, CLEAR TO SEND, DATA SET READY y DATA TERMINAL READY han de estar a nivel MARK ("1").

PIN 3 - RECEIVE DATA
Es por este pin por donde se recibe el TRANSMIT DATA enviado por otra unidad.

PIN 4 - REQUEST TO SEND
Esta señal se pone a nivel alto cuando el interface desea transmitir datos.

PIN 5 - CLEAR TO SEND
Indica que el data set o el MODEM está listo para transmitir. Esta señal es generada en respuesta de REQUEST TO SEND. En conexión con periféricos se conectará directamente a la señal REQUEST TO SEND.

PIN 6 - DATA SET READY
Indica que el data set o el MODEM está listo para enviar y/o recibir datos. Esta señal se unirá con el DATA TERMINAL READY en interface con periféricos.

PIN 7 - SIGNAL GROUND
A este pin se conectarán todas las masas lógicas.

PIN 8 - DATA CARRIER DETECT
Indica la presencia en la línea de la señal portadora (usualmente se utiliza para la comunicación vía MODEM).

PIN 9 - Reservado para DATA SET TESTING.

PIN 10 - Reservado para DATA SET TESTING.

PIN 12 - SECONDARY RECEIVE LINE DETECTOR.
Esta línea es el canal secundario y se utiliza en altas velocidades para transmitir mensajes o estatus o en dirección opuesta al flujo principal de datos. El canal secundario no transmite nunca en la misma dirección que el canal primario, al mismo tiempo.

PIN 13. SECONDARY CLEAR TO SEND.
Es lo mismo que CLEAR TO SEND pero en canal secundario. Ninguno de los canales denominados secundarios puede ser utilizado en sistemas funcionando a mes de 9600 baudios.

PIN 14 - SECONDARY TRANSMITED DATA
Igual que TRANSMIT DATA pero en canal secundario.

PIN 15 - TRANSMISION SIGNAL ELEMENT TIMING.
Es el clock de transmisión utilizado en comunicaciones SINCRONAS.

PIN 16 SECONDARY RECEIVED DATA.
Es lo mismo que RECEIVED DATA pero en canal secundario.

PIN 17 - RECEIVED SIGNAL ELEMENT TIMING.
Es el clock de transmisión recibido desde el exterior en comunicación SINCRONA.

PIN 19 - SECONDARY REQUEST TO SEND.
Es lo mismo que REQUEST TO SEND pero en canal secundario. Nótese que esta señal no aparece mientras la señal REQUEST TO SEND está a "ON".

PIN 20. DATA TERMINAL READY.
Esta señal se utiliza para indicar que el data set de este "device" está listo para recibir y/o (depende del REQUEST TO SEND) transmitir datos.

PIN 21. SIGNAL QUALITY DETECT.
Indica (cuando está OFF) que la portadora en la línea está en tal estado que probablemente habrá que repetir la transmisión.

PIN 22. RING INDICATOR
Se utiliza en MODEM's AUTO-ANSWER para indicar llamada por vía telefónica.

PIN 23 - DATA SIGNAL RATE DETECTION.
Cuando en la línea esta instalado un MODEM con dos velocidades, esta señal se utiliza para seleccionar una u otra.

PIN 24. TRANSMIT SIGNAL ELEMENT TIMING.
Por este pin se permite la entrada de un reloj externo para la velocidad de transmisión.

PIN 11. Señal sin asignar.
PIN 18. Señal sin asignar.
PIN 25. Señal sin asignar.
(Continuará).

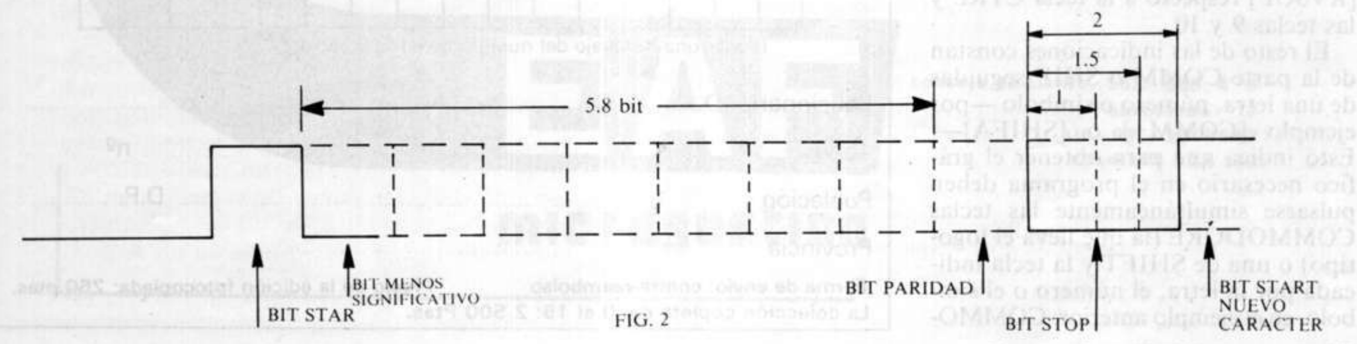



FIG. 2

# Clave para interpretar los listados de CLUB COMMODORE

Todos los listados que se publican en esta sección han sido ejecutados en el modelo correspondiente de la gama de ordenadores COMMODORE. Para facilitar la edición de los mismos en la sección y para mejorar su legibilidad por parte del usuario, se les ha sometido a ciertas modificaciones mediante un programa escrito especialmente para ello. Para los programas destinados a los ordenadores VIC-20 y COMMODORE 64, en los que se usan frecuentemente las posibilidades gráficas del teclado, se han sustituido los símbolos gráficos que aparecen normalmente en los listados por una serie de letras entre corchetes [] que indican la secuencia de teclas que se deben pulsar para obtener el carácter deseado. A continuación se da una tabla para aclarar la interpretación de las indicaciones entre corchetes:

[CRSRD] = Tecla cursor hacia abajo (sin SHIFT)

[CRSRU] = Tecla cursor hacia arriba (con SHIFT)

[CRSRR] = Tecla cursor a la derecha (sin SHIFT)

[CRSRL] = Tecla cursor a la izquierda (con SHIFT)

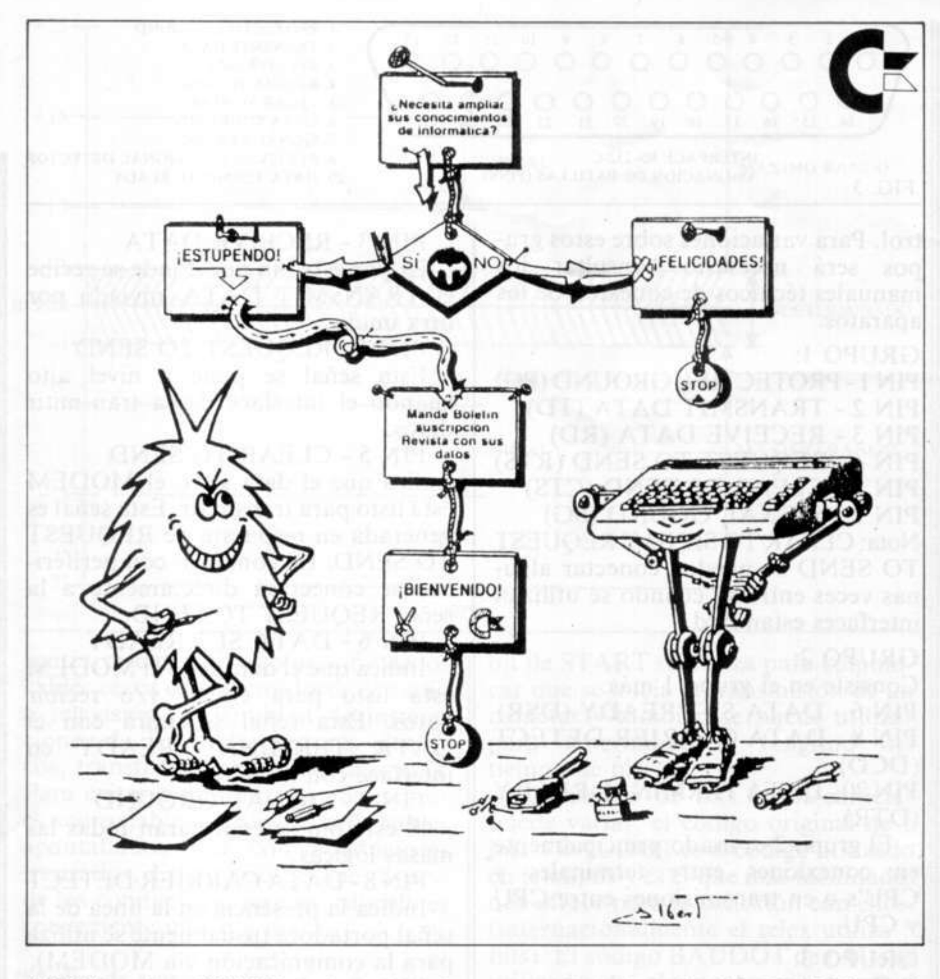
[HOME] = Tecla CLR/HOME (sin SHIFT)

[CLR] = Tecla CLR/HOME (con SHIFT)

Las indicaciones [BLK] a [YEL] corresponden a la pulsación de las teclas de 1 a 8 junto a la tecla CTRL. Lo mismo sucede con [RVSON] y [RVSOF] respecto a la tecla CTRL y las teclas 9 y 10.

El resto de las indicaciones constan de la parte COMM o SHIF seguidas de una letra, número o símbolo —por ejemplo [COMM+] o [SHIFA]—. Esto indica que para obtener el gráfico necesario en el programa deben pulsarse simultáneamente las teclas COMMODORE (la que lleva el logotipo) o una de SHIFT y la tecla indicada por la letra, el número o el símbolo, en el ejemplo anterior: COMMODORE y + o SHIFT y A, respectivamente.

En los signos gráficos además se cuenta el número de veces que aparece. Por ejemplo, [7 CRSRR] equivale a 7 pulsaciones de la tecla cursor a la derecha y [3 SPC] tres pulsaciones de la barra espaciadora. ■

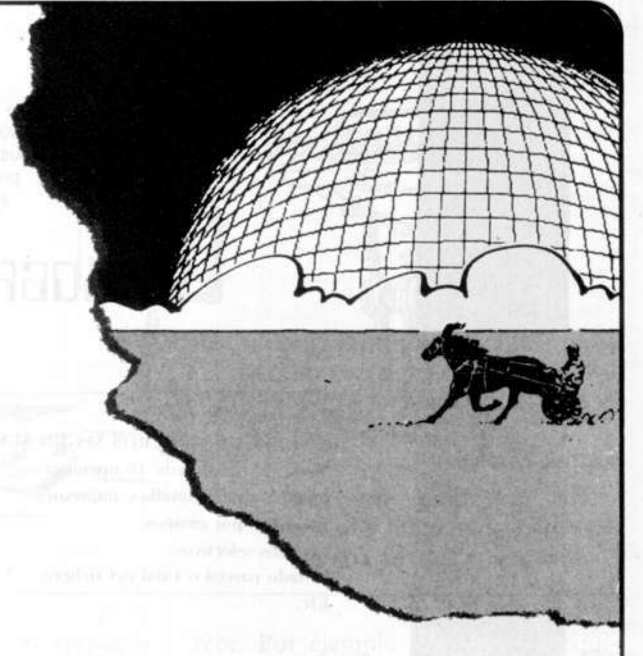




# Una Excursión en Basic
# Más allá del Manual

Por Jeffrey Mills

**Aquí presentamos un salvavidas para el entusiasta que se acaba de comprar un Commodore 64. Este es el primero en una serie de artículos que le ayudará a navegar por los rápidos y los remolinos de la programación del Basic.**



Así que te has leído el manual. Además, has programado los ejemplos del manual. ¿Pero cómo vas a encajar toda esta información? Vamos a ver algunas de las cosas que tú y tu Commodore 64 podéis hacer.

El primer paso en aprender a utilizar el ordenador de manera eficaz es el de saber programarlo para que satisfaga tus propias necesidades. ¿Y qué es un programa? Una definición simplificada podría ser: Un programa es un conjunto de instrucciones detalladas que le indican al ordenador, paso a paso, cómo realizar una serie de tareas, en un orden lógico, para poder alcanzar una meta.

"¿Pero cómo empiezo?" estarás diciendo. Pues, primero vamos a hablar de las normas básicas para la preparación previa y luego empezaremos a desarrollar un programa.

**Comandos para Operaciones Preparatorias**

1) *New (Nuevo)*—El comando "NEW" es muy potente, y se tiene que usar con cuidado. Cuando quieres iniciar un programa nuevo, "New" borra el programa anterior almacenado en la memoria, además de las variables para dicho programa. Más adelante consideraremos las variables.

Pero ojo. Una vez introducido el comando "NEW", no es posible recuperar el programa anterior, así que hay que salvarlo antes de utilizar este comando si no lo quieres perder. Para refrescar la memoria sobre New, consulta la página 115 en la "Guía para Usuarios del Commodore 64".

2) *CLR*—Este comando "despeja" la memoria del ordenador sin modificar ni borrar cualquier programa que se haya almacenado allí. Todas las variables se vuelven a poner a cero, y los punteros se vuelven a sus estados iniciales. (Los punteros son las direcciones de las diversas cosas importantes que el ordenador tiene que recordar mientras se está ejecutando un programa). El tema de CLR se trata en la página 117 de la Guía del Usuario.

3) *LIST (Listado)*—Este comando te permite ver un listado del programa que estás procesando. Para parar el comando "List", utiliza la tecla run/stop.

Puedes ejecutar "List" para que sólo aparezca una parte determinada. Esto se hace especificando los números de las líneas que quieres ver. Por ejemplo:

- **LIST 100-200 (se presentan todas las líneas de 100 hasta 200), o**
- **LIST -999 (se presentan todas las líneas hasta inclusive la línea 999), o**
- **LIST 345 (sólo se presenta la línea 345).**

Si se da el comando "List" sin especificar ningún número de línea, se presenta en listado el programa entero. El comando "List" se define con ejemplos en la página 115 en la Guía del Usuario.

4) *Line Numbering (Numeración de Líneas)*—Cuando se escribe un programa, el ordenador necesita saber el orden en que tiene que realizar las diversas tareas. Esto se realiza mediante los números de líneas, que pueden abarcar desde 0 hasta 65535.

Al numerar un programa, es mejor contar en decenas (10, 20, 30, 40...). Con este método se puede volver atrás para insertar una línea olvidada. Por ejemplo, podría resultar necesario insertar algo entre la primera línea (10) y la segunda (20). La nueva línea podría en este caso numerarse 15. Si las líneas se hubieran numerado 1, 2, 3, 4..., otra línea no se podía haber introducido entre 1 y 2.

5) *REM*—Este comando, que introduce un comentario explicativo, no afecta en absoluto la operación del programa, sino que aclara simplemente lo que el programa tiene que hacer en este punto. Esta documentación es bastante importante, ya que con el tiempo se olvida fácilmente lo que hay que hacer.

Lo primero que se aprende a utilizar en un programa real es la sentencia REM. (Los comandos que hemos explicado hasta ahora normalmente se introducen directamente a través del teclado). Las sentencias REM también constituyen la herramienta más útil para que el programador descubra la causa de mal funcionamiento de un programa. Por lo tanto, se deben usar con frecuencia para escribir un programa. A continuación, presentamos un ejemplo:

```
10 REM *** THIS IS A REMARK
   (ESTO ES UN COMENTARIO).
```

Toma nota del uso de los asteriscos, que se colocan en la sentencia para que ésta resulte más visible cuando un programa se presenta en forma de listado. Aunque no sean imprescindibles, son útiles. REM se explica en la página 124 de la Guía del Usuario.

**Los Primeros Pasos**

Lo primero que hay que hacer para desarrollar un programa es decidir exactamente qué es lo que tú quieres que realice el ordenador. El ejemplo que te voy a presentar indica cómo el Commodore 64 guarda el listado de todos los programas, incluyendo los números de las cintas donde se almacenan.

Empezamos con un comentario que identifica el programa y su autor. Se teclea:

```
10 REM *** PROGRAMA/CATA-
   LOGO DE CINTAS ***
20 REM *** ESCRITO POR: tu
   nombre **.
```

El segundo paso es el de despejar la pantalla, donde se va a representar el catálogo de programas. Esto se puede realizar desde dentro del programa utilizando un comando incluido en una sentencia PRINT.

La sentencia Print le indica al ordenador que tiene que representar algo directamente en pantalla, empezando a partir de la siguiente línea disponible. Si la pantalla está llena, el Commodore desplaza el texto hacia arriba para que la parte de abajo queda libre para incluir la línea que tiene que imprimirse.

Por lo tanto, la primera línea desaparece. Debido a esto, debes, de momento, limitar la lista a 15 líneas, para evitar este desplazamiento. Trataré este problema en otro artículo.

Para que se imprima un número en pantalla, se utiliza una línea como la siguiente:

```
30 PRINT 12
```

Cuando el ordenador llega a la línea 30 imprime el número 12 en pantalla. Para que se imprima una palabra se tiene que escribir entre comillas. Por ejemplo:

```
30 PRINT "DOCE"
```

El ordenador imprimirá la palabra DOCE en la pantalla al llegar a la línea 30.

Al igual que se imprime una palabra, se puede imprimir cualquiera de las teclas de comando del teclado. Estos se llaman caracteres de comando incluido. Cuando se imprimen, realizan las tareas que les han sido asignadas, como mover el cursor (que indica la posición donde se imprimirá la siguiente unidad de datos) o borrar la pantalla.

Para borrar la pantalla se teclea:

```
30 PRINT "(Shift-CLR/HOME)"
```

Si se escribe esa tecla entre comillas la pantalla no se borra en seguida. Lo que aparece es un carácter gráfico de "corazón" invertido. Su presencia borra la pantalla en el momento apropiado mientras se ejecuta el programa. (Cuando se intercala un comando, siempre se representa como un carácter invertido distinto para cada tecla).

En el manual no se tratan los comandos incluidos de teclado. Tardarás un poco en aprender los caracteres invertidos que representan las teclas de comando. Sin embargo, esto se aprende mediante la práctica.

Resulta útil saber que para introducir un programa (o una sentencia directa) mediante el teclado, se puede usar una interrogación para representar la palabra PRINT. Cuando se presenta el listado del programa, saldrá la palabra PRINT. (Existen varias palabras abreviadas que se presentan en una tabla en la página 130-131).

La sentencia PRINT se explica en el manual en las páginas 22-29 y 123-124.

**Encabezamiento y Espaciado**

Cuando el ordenador termina de borrar la pantalla, se puede imprimir un encabezamiento.

Existen varios métodos para centrar el encabezamiento. De momento, es mejor incluir espacios en la sentencia PRINT. Se teclea:

```
40 PRINT" CATALOGO DE CIN-
TAS"
```

Hay 14 espacios en una sentencia PRINT.

¿Por qué 14? El encabezamiento ocupa 12 espacios y cada línea ocupa 40 espacios en la pantalla. Quedan 28 espacios que se dividen en partes iguales entre los dos lados.

Para que quede una línea en blanco después del encabezamiento y antes de la lista del catálogo, se imprime una línea vacía:

```
50 PRINT
```

Antes de hablar del listado, es útil conocer algunos de los métodos más sencillos para que la representación en pantalla quede ordenada.

Existen dos formas sencillas para dejar espacios en la salida en pantalla: la coma y el punto y coma. Si se utiliza el punto y coma, la salida no tendrá ningún espacio entre un campo y el siguiente. (El campo es un término que se emplea para indicar un grupo de caracteres que forman parte de un registro entero. Por ejemplo, el número de cinta en el programa actual constituye un campo; el nombre del programa constituye otro. Los dos campos combinados forman el registro completo).

Si se imprime la salida con comas, quedará espaciada en la pantalla en lo que se llaman zonas de impresión. Las zonas de impresión del C-4 tienen una longitud de diez espacios. Cada campo empieza en la siguiente zona de impresión que queda disponible. A continuación se presentan un par de ejemplos.

Supongamos que hay dos programas que se tienen que incluir en el listado del catálogo. Se llaman el Juego 1 y el Juego 2, y se salvan en la cinta 101. Los dos programas salen en el catálogo mediante las sentencias de impresión. Se teclea:

```
60 PRINT "101"; "GAME 1" (Juego 1)
70 PRINT "101", "GAME 2" (Juego 2)
```

La línea 60 se imprime con un punto y coma mientras que la línea 70 se imprime con una coma.

Si se ejecuta el programa se observa la diferencia entre las dos líneas.

La primera no contiene ningún espacio entre los campos, mientras que la segunda se imprime en zonas individuales.

Si la entrada de "número" en la línea 70 hubiera contenido más de diez caracteres, el nombre del programa se habría imprimido al principio de la siguiente zona de impresión disponible (en este caso, la columna 20 en vez de la columna 10).

Ahora se teclea la línea 60 para que se modifique y se parezca a la línea 70:

```
60 PRINT "101". "GAME 1" (Juego 1)
```

Ambas líneas quedarán espaciadas de modo uniforme al ejecutarse el programa.

Ahora que sabes teclear una línea de programa con una sentencia PRINT diferente para cada entrada en el catálogo, puedes seguir y terminar tu lista.

# Ficheros en disco (5)

# Técnicas de Acceso Directo

**3. Distribución de los ficheros en disco:**

**Siguiendo el indice marcado en el artículo III de la serie, a continuación vamos a ver cómo se distribuyen los ficheros en el diskette, profundizando sobre lo visto en el artículo III, apartado 1.**

**En todos los discos existen tres tipos diferentes de bloques de control, que son:**

**A) CABECERA DEL DISKETTE (Directory Header)**

**B) BLOQUES DEL BAM**

**C) BLOQUES DEL DIRECTORIO**

**Las funciones del BAM y el directorio ya fueron descritas en el artículo anteriormente mencionado. Veamos pues para qué nos sirve la cabecera del diskette:**

**Manuel AMADO**

A) CABECERA DEL DISKETTE

Cuando se inicializa un diskette, o bien se empieza a trabajar en él sin inicializar (caso del 4040, 8250 y 8050, que se inicializan automáticamente) el DOS se sitúa siempre en un determinado sector del diskette, dependiendo este sector exclusivamente del modelo de floppy empleado. Esta cabecera contiene una serie de parámetros que identifican al diskette y en su caso los punteros necesarios con el primer bloque de BAM y del directorio para poder acceder y procesar en su caso la información contenida en el diskette.

En la fig. 1 podéis observar las cabeceras de diskette correspondiente a los floppys CBM: modelos 4040, 1540, 1541 y 8050.

Observamos la diferente estructura de las cabeceras de disco entre el 4040 (es igual en los 1540 y 1541) y el 8050/8250. En primer lugar, al sólo tener que controlar el 4040 683 bloques, es suficiente disponer de 139 bytes de BAM para controlar su disponibilidad. Por ello, en el 4040 se aprovecha el sector correspondiente a la pista 18 sector 0 para ubicar en él el BAM y la cabecera del disco. Por el contrario, en el caso del 8050 se necesitan controlar 2083 bloques, con lo que son necesarios dos bloques de BAM para controlarlos, y consecuentemente la cabecera del disco y los bloques de BAM están separados y en la cabecera del disco se hace necesario un puntero al primer sector del BAM.



**FIGURA 1**

**«DIRECTORY HEADER» 4040, 1540 y 1541**

**Pista 18, Sector 0**

| BYTE | CONTENIDO | DEFINICION |
|---|---|---|
| 144-161 | | Nombre del disco rellenado con espacios «shifted» |
| 162-163 | | ID del Disco |
| 164 | 160 | Espacios «shifted» |
| 165,166 | 50,65 | Representación ASCII para 2A que es la versión y tipo de formato del DOS |
| 166-167 | 160 | Espacios «shifted» |
| 171-255 | 0 | Nulos, no se usan |

**NOTA: Los caracteres ASCII pueden aparecer de la posición 180 a la 191 en algunos diskettes.**

**BLOQUE «DIRECTORY HEADER» DEL 8050**

**Pista 39, Sector 0**

| BYTE | CONTENIDO | DEFINICION |
|---|---|---|
| 0,1 | 38,0 | Pista y sector del primer bloque del BAM |
| 2 | 67 | El carácter ASCII C el formato 8050 |
| 3 | 0 | Señal nula para posteriores usos del DOS |
| 4,5 | 0 | No se usan |
| 6-21 | | Nombre del disco rellenado con espacios «shifted» |
| 22,23 | 160 | Espacios «shifted» |
| 24,25 | | I D del disco |
| 26 | 160 | Espacios «shifted» |
| 27,28 | 50,67 | Representación en ASCII del 2C que es versión y tipo de formato del DOS |
| 29-32 | 160 | Espacios «shifted» |
| 33-255 | 0 | No se usan |

La estructura en el modelo 8250 es análoga a la del 8050.

El resto de información contenida en las cabeceras se deduce de la simple observación de la figura 1.

## FIGURA 2
### FORMATO DEL BAM DEL 8050

**Primer bloque del BAM: Pista 38, Sector 0**

| BYTE | CONTENIDO | DEFINICION |
|---|---|---|
| 0,1 | 38,3 | Pista y sector del segundo bloque del BAM |
| 2 | 67 | Carácter ASCII C el formato del 8050 |
| 3 | 0 | Señal nula por futuros usos del DOS |
| 4 | 1 | El menor número de pista representado en este bloque del BAM |
| 5 | 51 | El mayor número de pista + 1 en este bloque del BAM |
| 6 | | Número de bloques no usados en la pista 1 |
| 7-10 | | Mapa de representación de bit de bloques disponibles en la pista 1 |
| 11-255 | | * BAM para las pistas del 2 al 50 5 bytes por pista |

**Segundo bloque del BAM: Pista 38, Sector 3**

| BYTE | CONTENIDO | DEFINICION |
|---|---|---|
| 0,1 | 39,1 | Pista y sector del primer bloque del directorio |
| 2 | 67 | Carácter ASCII C formato de 8050 |
| 3 | 0 | Señal nula para usos posteriores del DOS |
| 4 | 1 | El menor número de pista representado en este bloque BAM |
| 5 | 51 | El mayor número de pista + 1 en este bloque BAM |
| 6 | | Número de bloques no usados en la pista 51 |
| 7-10 | | Mapa de representación de bit de los bloques utilizables en la pista 51 |
| 11-140 | | * BAM para pistas 52-77, 5 bytes por pista |
| 141-255 | | No usado |

***ESTRUCTURA DE ENTRADA DE BAM PARA UNA PISTA**

| BYTE | DEFINICION |
|---|---|
| 0 | número de sectores disponibles para pista |
| 1 | mapa de bits de los sectores 0-7 |
| 2 | mapa de bits de los sectores 8-15 |
| 3 | mapa de bits de los sectores 16-23 |
| 4 | mapa de bits de los sectores 24-31 |

1 = disponible
0 = no disponible

**NOTA: «BLOCKS FREE» puede aparecer en las posiciones 180 a la 191 en algunos diskettes.**

## FIGURA 2
### FORMATO DEL BAM DEL 4040

**Pista 18, Sector 0**

| BYTE | CONTENIDO | DEFINICION |
|---|---|---|
| 0,1 | 18,01 | Pista y sector del primer bloque del directorio |
| 2 | 65 | El carácter ASCII A indicando el formato del 4040 |

### B) ESTRUCTURA DE LOS BLOQUES DEL BAM;

En el caso de la estructura de los bloques de BAM del 4040 y del 8050/8250, las diferencias apreciadas son notables con respecto al caso anterior (ver fig. 2). Una característica importante y común a ambos modelos es que los dos primeros bytes del último bloque de BAM contienen el puntero o link al primer bloque del directorio.

En el caso del 4040, la estructura en sí del BAM (lo que es el mapa de bits propiamente dicho), no tiene ninguna complicación, ya que cabe enteramente en la primera parte de la cabecera del disco y está distribuido por pesos, o sea, el bit 343 nos informa de la disponibilidad del sector o bloque número 343 del disco.

Para el 8050, se complica esta estructura ligeramente, ya que tiene que distribuir el BAM entre dos bloques del disco, 38-0 y 38-3. El primero controla los sectores comprendidos entre la pista 1 y 50, conteniendo en los dos primeros bytes un puntero a la pista y sector del siguiente bloque de BAM. Este segundo bloque del BAM contine en los dos primeros bytes el puntero o link al primer bloque del directorio, y controla el resto de las pistas del 8050 (51-77). Además, se repite en cada bloque el carácter 67 indicador de que se trata del formato 8050. La estructura de los bloques de BAM del 8250 es análoga a la del 8050, con la diferencia de que al tener que controlar 4133 bloques (prácticamente el doble del 8050), se necesitan 4 bloques de BAM.

### C) BLOQUES DEL DIRECTORIO

El directorio del disco está formado por los nombres y direcciones de los distintos ficheros que componen este disco. Su estructura es común a todos los floppys CBM, y cada bloque contiene en los dos primeros bytes la dirección del siguiente bloque del directorio, conteniendo 0 y 255 (0 y $FF en hexadecimal) en el caso de que sea el último bloque. El resto del bloque corresponde a las ocho entradas de fichero que puede llegar a contener dicho bloque.

En la figura 3 observamos que para cada fichero existe su entrada correspondiente en el directorio. Pues bien, el contenido de estos bytes que en la figura 3 se representa como "entrada fichero X", depende del tipo de fichero que describa.

Resumiendo, los bloques del directorio forman una estructura encadenada o lista, en la que cada uno de sus elementos apunta o está lincado al siguiente mediante un puntero o link. Estos bloques forman como una especie de índice de los ficheros del disco, pues nos indican en qué lugar del disco se halla cada fichero.

La estructura de la entrada de fichero del directorio se puede observar en la figura 4.

| | | |
|---|---|---|
| 3 | 0 | Señal nula para posteriores utilizaciones del DOS |
| 4-143 | | * Mapa de bit de bloques disponibles para las pistas 1-35. (BIT MAP) |

* 1 = bloque disponible
0 = bloque no disponible
cada bit representa un bloque.

## FIGURA 3
### FORMATO DEL DIRECTORIO

**Pista 18, Sector 1 para el 4040, 1540 y 1541**
**Pista 39, Sector 1 para el 8030**

| BYTE | DEFINICION |
|---|---|
| 0,1 | Pista y sector del siguiente bloque directorio |
| 2-31 | * Entrada de fichero 1 |
| 34-63 | * Entrada de fichero 2 |
| 66-95 | * Entrada de fichero 3 |
| 98-127 | * Entrada de fichero 4 |
| 130-159 | * Entrada de fichero 5 |
| 162-191 | * Entrada de fichero 6 |
| 194-223 | * Entrada de fichero 7 |
| 226-255 | * Entrada de fichero 8 |

* Ver Figura 4

## FIGURA 4
### ESTRUCTURA DE LA ENTRADA DEL DIRECTORIO SIMPLE

| BYTE | CONTENIDO | DEFINICION |
|---|---|---|
| 0 | 128 + tipo | Tipo de fichero OR'ed con $80 para indicar que el fichero ha sido correctamente cerrado. TIPOS: 0 = Borrado (DEL) 1 = Secuencial (SEQ) 2 = Programa (PRG) 3 = Usuario (USR) 4 = Relativo (REL) |
| 1,2 | | Pista y sector del primer bloque de datos |
| 3-18 | | Nombre del fichero rellenado con espacios «Shifted» |
| 19,20 | | Sólo ficheros relativos: pista y sector para el primer bloque side sector |
| 21 | | Sólo ficheros relativos: longitud de la ficha |
| 22-25 | | No usados |
| 26-27 | | Pista y sector del fichero de reemplazo cuando un OPEN @ entra en efecto |
| 28-29 | | Número de bloques en el fichero: menor byte, mayor byte (peso bajo, peso alto) |

## FIGURA 5
### FORMATO DE UN FICHERO DE PROGRAMA

| BYTE | DEFINICION |
|---|---|
| 0,1 | Pista y sector del siguiente bloque en el fichero de programa |
| 2-256 | 254 bytes de información del programa almacenado en el formato de memoria en el VIC. (Con las palabras clave bajo forma simbólica). El final de fichero está marcado con tres bytes iguales a 0 |

En ella hay el tipo de fichero que es, el puntero al primer bloque de datos del fichero, el nombre del fichero, el número de bloques que ocupa el fichero y una serie de parámetros propios de los ficheros relativos. La estructura de los bloques de datos para los ficheros secuenciales y programa puede verse en la figura 5. En el caso de ser un fichero programa, los dos primeros bytes de información del primer bloque del fichero contienen la dirección de carga en memoria de dicho programa.

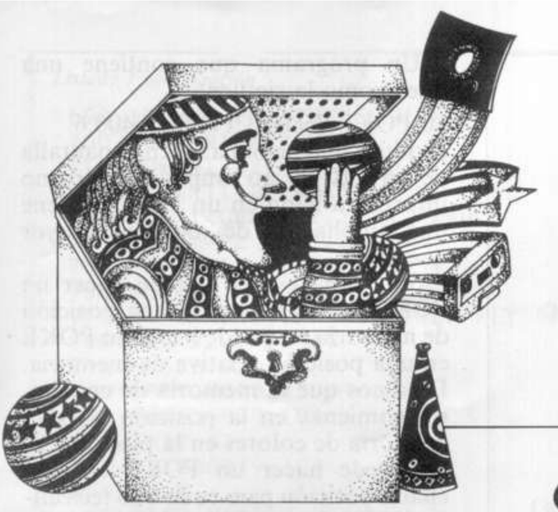
Consecuentemente, en los ficheros secuenciales y programa, su distribución es semejante en ambos casos: cada bloque de datos contiene en los dos primeros bytes el puntero al siguiente bloque (0, 255 si es el último), y los 254 bytes restantes contienen los bytes de datos o información propios del fichero. Observamos pues que ambos tienen una estructura parecida a la de los bloques del directorio, o sea, una lista o cadena de bloques.

Y ahora me haréis una buena pregunta: pues muy bien, muy bonito, pero, ¿para qué sirve toda esta disertación? Pues es bien sencillo y complicado a la vez de contestar. Para varias cosas, y sobre todo dependiendo de la imaginación que le pongáis al asunto. Vamos a ver solamente dos aplicaciones sencillas: (os apunto la idea, más adelante se plasmarán en programas, o si os animáis y escribís a la redacción los programas basados en alguna de estas dos ideas, se publicarán las mejores versiones de cada idea).

A) Supongamos que se ha estropeado un determinado bloque del disco, pues nos da al acceder a él un error 23, READ ERROR, por ejemplo, al leer los datos en un fichero secuencial. Pues bien, podemos intentar recuperar parte de los datos del fichero siguiendo la cadena de ficheros encadenados por acceso directo, y marcando el anterior bloque al que da error como fin del fichero, o bien, si localizamos el bloque siguiente (suponiendo que no esté estropeado) al bloque que nos da error, podemos reconstruir la cadena de bloques linkados haciendo un puente entre el anterior bloque al erróneo y el posterior, con lo cual sólo habremos perdido un bloque de información del fichero.

B) Hacer un utilitario de copia selectiva de ficheros de un diskette a otro del mismo floppy, mediante exploración del contenido del directorio del diskette a copiar, y seleccionando el operador sólo aquellos ficheros que se deseen copiar.

Para finalizar deseo hacer constar que la información sobre los ficheros relativos fue publicada en los números 9, 10 y 11 de nuestra revista. ■





# Video Casino

*Cada mes,* COMMODORE WORLD, *presentará programas de juegos originales que vd. puede usar y disfrutar con su micro. El primero de la serie se titula "Tiro al Blanco".*

La galería de tiro al blanco ha sido, y sigue siendo, un juego muy popular. Requiere la coordinación entre la mano y el ojo en contra de los caprichos del blanco que se mueve al azar.

Algunos de los primeros juegos para el ordenador eran adaptaciones de los muchos juegos de tiro al blanco. A veces el contrincante era un vaquero en video. Hoy en día, los blancos pueden ser naves que navegan sobre la superficie de un océano en la pantalla, por ejemplo, o invasores peligrosos de otra galaxia.

"Tiro al Blanco" presenta un blanco que se mueve rápidamente. Una caja se desplaza por la parte superior de la pantalla de izquierda a derecha. Al pasar, va descendiendo cada vez más, hasta que al final, el blanco choca con su nave localizada en la parte inferior de la pantalla. Vd. puede usar el mando (joystick) para mover la nave de un lado para otro, para seguir el paso del blanco si así se desea. En el momento más oportuno, al pulsar el botón de disparo, se lanza un misil.

En este momento, la nave no se mueve, y el mando (joystick) controla el misil. Vd. acerca el proyectil al blanco enemigo. Si acierta en el blanco, se desencadena una explosión impresionante. Si lo hace con cuidado puede acertar dos, tres o más veces en una sola pasada, acumulando una puntuación realmente alta.

**Variable Utiles**

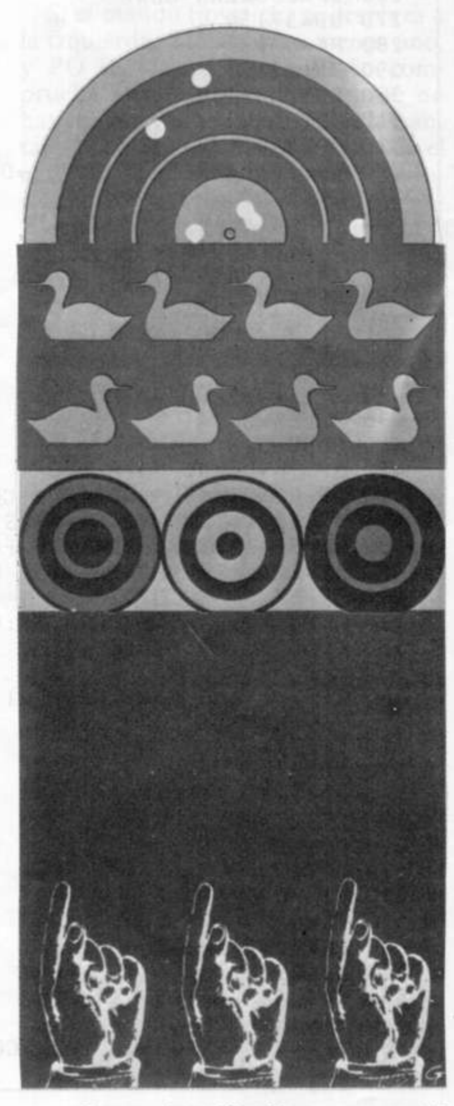
El programa está dividido en varios módulos. El primer paso después de las instrucciones es el de inicializar ciertas variables que se utilizarán más adelante en el programa. Al definir números que se utilizarán una y otra vez como variables, la operación del programa se hará mucho más rápida. En vez de evaluar la constante 6, por ejemplo, cada vez que el programa la encuentra, el ordenador utiliza la variable BLUE (azul).

El ordenador comprueba esta variable en una tabla interna rápidamente, dado que todas las variables se almacenan en el orden en que el programa las utiliza. Dado que BLUE fue definido en la línea 200, se encontrará al principio de la tabla de verificación de variables. En vez de hacer un POKE en una posición en la memoria de color con el 6 para que se vuelva azul, se puede hacer un POKE de la variable BLUE, que es más fácil de recordar que el número seis.

De forma parecida, en vez de hacer un POKE 32 en la memoria de pantalla para conseguir un espacio (CHR$ (32)), simplemente se hace un POKE de la variable SPACE, definida en la línea 230. También se definen dos variables adicionales, CSCREEN y CHAR, pero debido a otros motivos.

Como vd. bien sabe, el VIC-20 almacena un registro de lo que aparece en la pantalla en una serie de 506 posiciones de memoria. El color del carácter de cada una de dichas posiciones también se almacena en 506 posiciones de memoria. Si se cambia el contenido de cualquiera de estos bytes se cambia el carácter en dicha posición de pantalla, o se cambia su color.

En el VIC-20 no ampliado, los códigos para los caracteres de pantalla se almacenan a partir de la posición de memoria 7680 y el mapa de memoria del código de color comienza a partir del 38400. Desafortunadamente, estas posiciones de memoria pueden cambiar, según la cantidad de memoria añadida al VIC-20. Por eso muchos de los juegos en venta especifican que se usan solamente con un VIC-20 no ampliado.



*El Listado 1. Programa para "Tiro al Blanco" en el VIC-20.*

```
1 REM VIC-20 VERSION
10 REM ****************
20 REM *              *
30 REM *TIRO AL BLANCO*
40 REM *              *
50 REM ****************
60 PRINT"[SHFT CLR][CRSR DN][CRSR DN]"
70 PRINTTAB(4)"[CTRL 9][CTRL 3]TARGET SHOOT[CRSR DN][C
   RSR DN]"
80 PRINTTAB(1)"USE JOYSTICK TO MOVE"
90 PRINTTAB(1)"YOUR BASE ON BOTTOM"
100 PRINTTAB(1)"OF SCREEN, AND TO "
110 PRINTTAB(1)"STEER YOUR ARROW.TRY"
120 PRINTTAB(1)"TO GET AS MANY HITS"
130 PRINTTAB(1)"AS POSSIBLE BEFORE"
140 PRINTTAB(1)"TARGET REACHES BOTTOM[CRSR DN][CRSR DN]
   "
150 PRINTTAB(6)"[CTRL 9][CTRL 3]HIT ANY KEY"
160 GET A$:IF A$="" GOTO 160
170 POKE36879,104
180 PRINT"[SHFT CLR]"
190 DF=30720
200 WHITE=1:CYAN=3:BLUE=6
210 RSPACE=160
220 ARROW=30
230 SPACE=32
240 CSCREEN=37888+4*(PEEK(36866)AND128):B1=CSCREEN
250 CHAR=4*(PEEK(36866)AND128)+64*(PEEK(36869)AND120):B
   =CHAR:E=CHAR+484
260 DF=CSCREEN-CHAR
270 DD=37154
280 PA=37137
290 PB=37152
300 POKE 37139,0
310 GOTO 390
320 POKEDD,127
330 S3=-((PEEK(PB)AND128)=0)
340 POKE DD,255
350 P=PEEK(PA)
360 S2=((PAND16)=0)
370 FR=-((PAND32)=0)
380 RETURN
390 PRINT"[SHFT CLR]"
400 SH=CSCREEN
410 PO=E
420 HITS=0:BALLS=0
430 BALLS=0
440 GOSUB320
450 IF SH-DF>PO GOTO 870
460 IF S3=0 AND S2=0 THEN GOTO 530
470 IF F2=1 THEN N1=N1+S2+S3:GOTO530
480 IF S3=1THENPO=PO+1:IF PO>E+22THENPO=E+22
490 IF S2=-1THENPO=PO-1:IF PO<ETHENPO=E
500 IF S2=-1 GOTO 520
510 POKE PO,65:POKEPO-1,32:GOTO530
520 POKE PO,65:POKEPO+1,32
530 SH=SH+1
540 IFF2=0 AND FR=1THENFL=1:LE=50:GOSUB1020
550 IF FLAG=1THENGOSUB620
560 IF F2=1THENGOSUB670
570 POKE SH,CYAN
580 POKESH-DF,RSPACE
590 POKE SH-1,BLUE
600 POKESH-DF-1,32
610 GOTO 440
620 N1=PO-22:F2=1
630 FL=0
640 POKE N1,ARROW
650 N1=N1-22
660 RETURN
670 H=PEEK(N1)
680 IFH=81THEN GOSUB 1020:GOTO 700
```

Sigue

Un programa que contiene una línea como la siguiente:
10 POKE 7680,81:POKE38400,6
produce una pelota azul en la pantalla de un VIC-20 no ampliado, pero no imprimirá nada en un VIC que tiene una ampliación de memoria mayor que el 3K.

Sin embargo, en vez de hacer un POKE directamente en una posición de memoria, se puede hacer un POKE en una posición relativa de memoria. Digamos que la memoria de caracteres comienza en la posición X, y la memoria de colores en la posición Y. Se puede hacer un POKE en una cuarta posición para cada uno tecleando POKE X+4 o POKE Y+4. Si definimos X e Y como los diferentes puntos de partida, según la cantidad de memoria, el mismo programa funcionará en cualquier VIC-20.

Esto es lo que se hace en "Tiro al Blanco". En vez de X e Y, usamos CSCREEN (para la memoria de colores) y CHAR (para la memoria de caracteres). Las líneas 240 y 250 hacen un PEEK en una posición que les indica la memoria disponible que les queda, y luego calculan las verdaderas posiciones para los respectivos mapas de memoria.

Para simplificar un poco más, se calcula la diferencia (DF) entre los dos puntos de partida. Así, si queremos hacer un POKE a la cuarta posición de cada mapa de memoria, simplemente tecleamos POKE CSCREEN + 4 y POKE CSCREEN + 4 - DF.

El motivo de hacer esto, por ejemplo, ès que la posición de la nave (definida como SH) cambia a medida que el jugador manipula el mando (joystick). Tanto el carácter como su color se cambian en la nueva posición al hacer un POKE en los nuevos valores de SH y SH-DF con el carácter y el color.

## EQUIVALENTES ESPAÑOL

1 REM VERSION VIC-20
10 -50 REM TIRO AL BLANCO
80 PRINTTAB(1) "UTILIZAR MANDO (JOYSTICK) PARA MOVER"
90 PRINTTAB(1) "TU BASE EN LA PARTE INFERIOR"
100 PRINTTAB(1) "DE LA PANTALLA, Y PARA"
110 PRINTTAB(1) "GUIAR TU FLECHA. INTENTA"
120 PRINTTAB(1) "ACERTAR LO MAS"
130 PRINTTAB(1) "POSIBLE ANTES DE QUE"
140 PRINTTAB(1) "BLANCO LLEGUE ABAJO (CRSR DN) (CRSR DN)"
150 HIT ANY KEY = "PULSE CUALQUIER TECLA"
950 NEW HIGH SCORE = "NUEVA MAXIMA PUNTUACION"
970 PLAY AGAIN? = "JUEGAS OTRA VEZ?"

*Listado 1 continuación*

```
690 IF H<>SPACETHENGOSUB770
700 POKE N1,ARROW
710 POKEN1+22,SPACE
720 POKEN11+DF,WHITE
730 IF N1<CHAR+22 THEN F2=0
740 N1=N1-22
750 POKE N1+22,SPACE
760 RETURN
770 H=PEEK(N1):IFH=81 THEN RETURN
780 FOR N2=1 TO 50
790 GOSUB1020
800 POKE36879,25
810 POKE36879,104
820 NEXT N2
830 POKE36879,104
840 F2=0
850 HITS=HITS+1
860 RETURN
870 POKE7910,81
880 POKE 38630,4
890 PRINT"[SHFT CLR]"
900 PRINT"[CTRL 8][SHFT Q][SHFT Q][SHFT Q][SHFT Q][SHFT
     Q][SHFT Q][SHFT Q][SHFT Q][SHFT Q][SHFT Q][SHFT Q
    ][SHFT Q][SHFT Q][SHFT Q][SHFT Q][SHFT Q][SHFT Q][
    SHFT Q][SHFT Q][SHFT Q][SHFT Q][SHFT Q]"
910 PRINT"          [CTRL 2]GAME OVER"
920 PRINT"[CTRL8][SHFT Q][SHFT Q][SHFT Q][SHFT Q][SHFT
    Q][SHFT Q][SHFT Q][SHFT Q][SHFT Q][SHFT Q][SHFT Q]
    [SHFT Q][SHFT Q][SHFT Q][SHFT Q][SHFT Q][SHFT Q][S
    HFT Q][SHFT Q][SHFT Q][SHFT Q][SHFT Q][3CRSR DNS]"
930 PRINT" HITS:      ";HITS
940 PRINT"[CRSR DN] SCORE :";HITS
950 IF HITS>HSTHENHI=SCR:PRINT"[CTRL 6] [CRSR DN]NEW HI
    GH SCORE![CTRL 1]":HS=HITS
960 PRINT"[CRSR DN] HIGH:";HS
970 PRINT"[CRSR DN]          [CTRL 9][CTRL 8]PLAY AGAIN?"
980 GET A$:IF A$=""GOTO980
990 IF A$="N"THEN END
1000 PRINT"[SHFT CLR]":GOTO390
1010 GOTO 390
1020 POKE36878,15
1030 POKE 36876,255
1040 FOR G1=1 TO LE
1050 NEXT G1
1060 POKE 36878,0
1070 LE=0
1080 RETURN
```

*El Listado 2. Programa de "Tiro al Blanco" para el C-64.*

```
10 REM ****************
20 REM *              *
30 REM * TARGET SHOOT *
40 REM *     C-64     *
50 REM ****************
60 VOLUME=54296
70 VCE=54273
80 WAVE=54276
90 ATTACK=54277
100 SUS=54278
110 POKE 53281,14
120 FOR N=1 TO 10
130 READ R:JV(N)=R
140 NEXT N
150 DATA 0,0,0,-1,0,0,0,1,0,0
160 PRINT"[SHFT CLR][CRSR DN][CRSR DN]"
170 PRINTTAB(12)"[CTRL 9][CTRL 3]TARGET SHOOT[CRSR DN]
    [CRSR DN]"
180 PRINTTAB(8)"USE JOYSTICK TO MOVE"
```

Sigue

En este artículo no entraremos en los detalles de cómo el VIC-20 lee los mandos (joystick). Dejaremos este tema para otro artículo. Sin embargo, el módulo que realiza esta función se encuentra en las líneas 320 a 380. La rutina devuelve valores para las variables S2, S3 y FR. Las dos primeras indican si el mando (joystick) se encuentra a la izquierda o a la derecha. Se hace caso omiso del movimiento hacia arriba o hacia abajo.

En este caso, se eligió un mando (joystick) en vez de un mando (paddle) (que el VIC-20 también acepta) dado que el alcance de control es mucho más preciso con el mando (joystick). Los mandos (paddles) hacen que el objeto se mueva más rápido, pero se necesita más precisión para colocar el objeto en la pantalla. La tercera variable, FR, le indica al programa si se ha pulsado o no el botón de disparo.

El movimiento de la nave es muy fácil. Si el mando (joystick) se desplaza a la derecha, S3 es igual a uno y la posición de la nave, PO, se aumenta por uno en la línea 480. Se comprueba que la base no se haya desplazado más allá del final (E) de la pantalla.

Si el mando (joystick) se desplaza a la izquierda, S2 es igual a menos uno, y PO se reduce por uno. Se comprueba una vez más que la nave no haya salido por la izquierda de la pantalla. Con PO se hace un POKE con el carácter de la nave, 65, y con su posición anterior (bien PO + 1 o bien PO - 1, según la dirección de movimiento) se hace un POKE con un espacio, 32 (space, 32).

A continuación, la posición del blanco, SH, se incrementa por uno (el blanco siempre se desplaza de izquierda a derecha), y con un espacio invertido hace un POKE en su nueva posición. Con la posición anterior se hace un POKE con un espacio.

Cuando se pulsa el botón de disparo se lanza un misil, ARROW, a la posición N1, mediante el POKE conveniente. Ahora el movimiento del mando (joystick) controla el valor de N1 a la izquierda o a la derecha. A cada "giro", sin embargo N1 se reduce por 22 para que ascienda de una fila en la pantala.

El programa hace PEEKS de vez en cuando para saber si se encuentra algo en el espacio que va a ocupar el misil. Si no es igual a un espacio significa que ha habido un choque, y el control bifurca a una rutina que provoca la explosión e incrementa la puntuación, HIT, por uno.

Cuando el misil llega a la parte más alta de la pantalla, el jugador se vuelve a hacer con el control de la nave y repite el ciclo, hasta que el blanco llega a la parte inferior de la pantalla, y termina el juego.

*Listado 2 Continuación*

```
190 PRINTTAB(8)"YOUR BASE ON BOTTOM"
200 PRINTTAB(8)"OF SCREEN, AND TO "
210 PRINTTAB(8)"STEER YOUR ARROW.TRY"
220 PRINTTAB(8)"TO GET AS MANY HITS"
230 PRINTTAB(8)"AS POSSIBLE BEFORE"
240 PRINTTAB(8)"TARGET REACHES BOTTOM[CRSR DN][CRSR DN]
   "
250 PRINTTAB(8)"[CRSR DN][CRSR DN]PLUG JOYSTICK INTO PO
   RT"
260 PRINTTAB(8)"CLOSEST TO POWER SWITCH.[CRSR DN][CRSR
   DN]"
270 PRINTTAB(12)"[CTRL 9][CTRL 3]HIT ANY KEY"
280 GET A$:IF A$="" GOTO 280
290 POKE 53280,0:POKE 53281,6
300 PRINT"[SHFT CLR]"
310 WHITE=1:CYAN=3:BLUE=6
320 RSPACE=160
330 ARROW=30
340 SPACE=32
350 CSCREEN=55296:B1=CSCREEN
360 CHAR=1024:B=CHAR:E=1983
370 DF=CSCREEN-CHAR
380 GOTO 430
390 JV=PEEK(56320)
400 FR=JVAND16
410 JV=15-(JVAND15)
420 RETURN
430 PRINT"[SHFT CLR]"
440 SH=CSCREEN
450 PO=E+20
460 HITS=0
470 F2=0
480 GOSUB390
490 IF SH-DF>PO GOTO 910
500 MOVE=JV(JV)
510 IF F2=1 THEN N1=N1+MOVE:GOTO560
520 PO=PO+MOVE:IF PO>E+40 THEN PO=E+40
530 IF PO<ETHENPO=E
540 POKE PO,65:POKEPO-1,32:POKE PO+1,32
550 POKE PO+DF,1
560 SH=SH+1
570 IFF2=0 AND FR=0THENFL=1:LE=50:GOSUB1030
580 IF FLAG=1THENGOSUB650
590 IF F2=1THENGOSUB700
600 POKE SH,CYAN
610 POKESH-DF,RSPACE
620 POKE SH-1,BLUE
630 POKESH-DF-1,32
640 GOTO 480
650 N1=PO-40:F2=1
660 FL=0
670 POKE N1,ARROW
680 N1=N1-40
690 RETURN
700 H=PEEK(N1)
710 IFH=81THEN GOSUB 1030:GOTO 730
720 IF H<>SPACETHENGOSUB800
730 POKE N1,ARROW
740 POKEN1+40,SPACE
750 POKEN11+DF,WHITE
760 IF N1<CHAR+40 THEN F2=0
770 N1=N1-40
780 POKE N1+40,SPACE
790 RETURN
800 H=PEEK(N1):IFH=81 THEN RETURN
810 FOR N2=1 TO 50
820 GOSUB1030
830 POKE 53280,1
840 POKE 53280,0
850 NEXT N2
860 POKE 53280,0
870 POKE 53281,6
880 F2=0
890 HITS=HITS+1
900 RETURN
910 PRINT"[SHFT CLR]"
920 PRINT"[CTRL 8][SHFT Q][SHFT Q][SHFT Q][SHFT Q][SHFT
   Q][SHFT Q][SHFT Q][SHFT Q][SHFT Q][SHFT Q][SHFT Q
   ][SHFT Q][SHFT Q][SHFT Q][SHFT Q][SHFT Q][SHFT Q][
   SHFT Q][SHFT Q][SHFT Q][SHFT Q][SHFT Q][SHFT Q][SH
   [SHFT Q][SHFT Q][SHFT Q][SHFT Q][SHFT Q][SHFT Q][S
   HFT Q][SHFT Q][SHFT Q][SHFT Q][SHFT Q][SHFT Q]"
930 PRINTTAB(12)"[CTRL 2]GAME OVER[CRSR DN]"
940 PRINT"[CTRL 8][SHFT Q][SHFT Q][SHFT Q][SHFT Q][SHFT
   Q][SHFT Q][SHFT Q][SHFT Q][SHFT Q][SHFT Q][SHFT Q
   ][SHFT Q][SHFT Q][SHFT Q][SHFT Q][SHFT Q][SHFT Q][
   SHFT Q][SHFT Q][SHFT Q][SHFT Q][SHFT Q][SHFT Q][SH
   [SHFT Q][SHFT Q][SHFT Q][SHFT Q][SHFT Q][SHFT Q][S
   HFT Q][SHFT Q][SHFT Q][SHFT Q][SHFT Q][SHFT Q][CRS
   R DN][CRSR DN][CRSR DN]"
950 PRINTTAB(8)"HITS:      ";HITS
960 PRINTTAB(8)"[CRSR DN]SCORE :";HITS
970 IF HITS>HSTHENHI=SCR:PRINTTAB(8)"[CTRL 6] [CRSR DN]
   NEW HIGH SCORE![CTRL 1]":HS=HITS
980 PRINTTAB(8)"[CRSR DN] HIGH:";HS
990 PRINTTAB(12)"[CRSR DN][CTRL 9][CTRL 8]PLAY AGAIN?"
1000 GET A$:IF A$=""GOTO1000
1010 IFA$="N"THEN END
1020 PRINT"[SHFT CLR]":GOTO430
1030 POKE VOLUME,15
1040 POKE WAVE,33
1050 POKE ATTACK,128
1060 POKE SUS,128
1070 POKE VCE,40
1080 POKE VOLUME,0
1090 RETURN                    READY.
```

## EQUIVALENTES ESPAÑOL

10 - 50 REM TIRO AL BLANCO C-64
180 PRINTTAB(8) "UTILIZAR MANDO (JOYSTICK) PARA MOVER"
190 PRINTTAB(8) "TU BASE EN LA PARTE INFERIOR"
200 PRINTTAB(8) "DE LA PANTALLA, Y PARA"
210 PRINTTAB(8) "GUIAR TU MISIL. INTENTA"
220 PRINTTAB(8) "ACERTAR LO MAS"
230 PRINTTAB(8) "POSIBLE ANTES DE QUE"
240 PRINTTAB(8) "BLANCO LLEGUE ABAJO (CRSR DN) (CRSR DN)"
250 PLUG JOYSTICK INTO PORT = "ENCHUFAR MANDO (JOYSTICK) EN SU CONECTOR"
260 CLOSEST TO POWER SWITCH = "MAS CERCANA A FUENTE DE ENERGIA"

270 HIT ANY KEY = "PULSE CUALQUIER TECLA"
980 HIGH = "MAXIMO"
990 PLAY AGAIN? = "VUELVES A JUGAR?"

# DESCRIPCIÓN ALFABÉTICA DE LOS MNEMÓNICOS DEL 6502/6510 (II)

## BCS — BCS

*Salta si acarreo activado*

Operación: Salta si C = 1

(Ref.: 4.1.1.4)

N Z C I D V
\- - - - - -

| Modo de Direc. | Formato en ensamblador | Código Operan. | Núm. Bytes | Núm. Ciclos |
|---|---|---|---|---|
| RELATIVO | BCS Oper. | B0 | 2 | 2* |

* Suma 1 si se salta a la misma página.
* Suma 2 si se salta a otra página.

## BIT — BIT

*Compara los bits de memoria con acumulador*

Operación: $A \wedge M$, M7 → N, $M_6 \rightarrow V$
Los bits 6 y 7 se transfieren al registro de estado. Si el resultado de $A \wedge M$ es cero entonces Z = 1, sino Z = 0.

(Ref.: 4.2.1.1)

N Z C I D V
M7* √ - - $M_6$

| Modo de Direc. | Formato en ensamblador | Código Operan. | Núm. Bytes | Núm. Ciclos |
|---|---|---|---|---|
| Pág. Cero | BIT Oper. | 24 | 2 | 3 |
| Absoluto | BIT Oper. | 2C | 3 | 4 |

## BEQ — BEQ

*Salta si el resultado es cero*

Operación: Salta si Z = 1

(Ref.: 4.1.1.5)

N Z C I D V
\- - - - - -

| Modo de Direc. | Formato en ensamblador | Código Operan. | Núm. Bytes | Núm. Ciclos |
|---|---|---|---|---|
| RELATIVO | BEQ Oper. | F0 | 2 | 2* |

* Suma 1 si se salta a la misma página.
* Suma 2 si se salta a otra página.

## BMI — BMI

*Salta si el resultado es negativo*

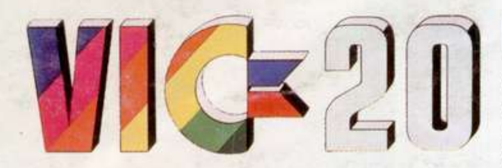
Operación: Salta si N = 1

(Ref.: 4.1.1.1)

N Z C I D V
\- - - - - -

| Modo de Direc. | Formato en ensamblador | Código Operan. | Núm. Bytes | Núm. Ciclos |
|---|---|---|---|---|
| Relativo | BMI Oper. | 30 | 2 | 2* |

* Suma 1 si se salta a la misma página.
* Suma 2 si se salta a otra página.




**Microprocesador:** 6502 de MOS TECHNOLOGY de 8 bits.
**Memoria:** 5 Kbytes de RAM ampliables a 32 K 20 Kbytes de ROM ampliables a 28 K.
**Pantalla:** 23 líneas de 22 caracteres. Modulador para conectar a un televisor normal. Salida monitor video.
**Colores:** 8 para el marco, 16 para el fonde de la pantalla y 8 para los caracteres individuales, video inverso.
**Gráficos:** Semi-gráficos por teclado y alta resolución por redefinición del generador de caracteres (situándolo en RAM). Definición de 176 por 184 puntos.
**Teclado:** Tipo QWERTY de 62 teclas más cuatro de función definibles por el usuario.
**Sonido:** Tres voces de tres octavas cada una decaladas una octava entre sí, resultando una extensión total de cinco octavas. Un generador de ruido aleatorio afinable para efectos especiales, un control general de volumen.
**Programación:** Lenguaje BASIC, intérprete residente en ROM de 8K. Posibilidad de interceptar las funciones del Basic para crear nuevas instrucciones "a medida". El Basic del Vic es uno de los rápidos actualmente en el mercado.
**Complementos:** Port de usuario de 8 bits entrada/salida más dos señales de sincronismo.
Bus de expansión para ampliaciones de memoria y periféricos.
Port de juegos con conexión para dos potenciómetros (paddles), y una palanca de juegos (joystick).
**Almacenamiento de masa:** Unidad de cassette C2N de diseño especial para registrar programas y datos.
**Ampliación de memoria:** En caso de ser necesario conectar más de un cartucho al mismo tiempo, está disponible un módulo (VIC 1020) que permite la conexión simultánea de hasta seis cartuchos.

## VIC-1541 UNIDAD DE DISCO

**Capacidad total:** 174848 bytes por disco.
**Secuencial:** 168656 bytes por disco.
**Entradas de directorio:** 144 por disco.
**Sectores por pista:** De 17 a 21.
**Bytes por sector:** 256.
**Pistas:** 35.
**Bloques:** 683 (644 bloques libres).
**Soportes de información:** Discos standar de 5 1/4 pulgadas, de una sola cara y densidad simple.
**Sistema operativo:** DOS de COMMODORE inteligente (tiene procesador propio y no ocupa memoria del ordenador central).

## VIC-1525 IMPRESORA

**Método de impresión:** Matriz de 5 × 7 puntos, impacto por un solo martillo.
**Modo caracteres:** Mayúsculas y minúsculas, símbolos, números y caracteres gráficos del VIC-20.
**Modo gráfico:** Puntos direccionables (bit image). Siete puntos verticales por columna, 480 columna máximo.
**Velocidad:** 30 caracteres/segundo, de izquierda a derecha, unidireccional.
**Caracteres/Línea:** Máximo 80. (Posibilidad de impresión en doble ancho).
**Espaciado entre líneas:** 6 líneas/pulgada —modo caracteres, 9 lineas/pulgadas— modo gráfico.
**Alimentación de papel:** Arrastre por tractor.
**Ancho de papel:** Entre 4,5 y 10 pulgadas.
**Copias:** Original más dos copias.

## CARTUCHOS

**Ayuda programador:** Facilita la edición y depuración de programas en Basic. Instrucciones y comandos: RENUMBER, MERGE, FIND, CHANGE, DELETE, AUTO, TRACE, STEP, OFF, KEY, EDIT, PROG, DUMP, HELP y KILL.
**Super expander:** Intercepta el Basic del VIC permitiendo incrementar sus instrucciones y comandos en aplicaciones gráficas de sonido y juegos. Instrucciones y comandos: KEY, GRAPHIC, COLOR, POINT, REGION, DRAW, CIRCLE, PAINT, CHAR, SCNCLR, SOUND, RGE, RCOLR, RDOT, RPOT, RPEN, RJOY y RSND.
**Monitor de lenguaje máquina:** Facilita enormemente la depuración de programas en lenguaje máquina, es ideal como complemento del Basic para redactar y poner en marcha rutinas de alta velocidad y manejo de datos en tiempo real. Instrucciones y comandos: ASSEMBLE, BREAKPOINT, DISASSEMBLE, ENABLE, VIRTUAL ZERO PAGE, FILL MEMORY, GO, HUNT, INTERPRET, JUMP TO SUBROUTINE, LOAD, MEMORY, NUMBER, QUICK TRACE, REGISTERS, REMOVE BREACPOINTS, SAVE, TRANSFER, WALK y EXIT TO BASIC.
Además existen cartuchos de ampliación de memoria de 3,8 y 16 Kbytes.

## CURSO DE INTRODUCCION AL BASIC PARTE I y II.

En forma de libro se ha editado la primera y segunda parte de un curso de Basic que parte "de cero" y está basado en el VIC-20. Van acompañados de dos cassettes con programas y ejercicios para autocontrol.

## PLOTTER VIC 1520

**Método de impresión:** Dibujo mediante bolígrafos de diseño especial.
**Color:** 4 colores; negro, azul, verde y rojo con cambio desde programa.
**Cabezal:** Ploter X-Y tipo tambor.
**Velocidad de impresión:** Media de 14 car./seg.
**Caracteres por línea:** Máximo 80 carac., formatos de 80, 40, 20 y 10 carac./linea.
**Juego de caracteres:** 96.
**Velocidad de dibujo:** 264 pasos/seg.
**Longitud del paso:** 0,2 mm. en dirección X e Y.
**Velocidad de dibujo de línea:** 52,8 mm./seg. en dirección X e Y. 73 mm./seg. en una línea a 45 grados.
**Area de dibujo:** 480 pasos (96 mm.) en dirección X. Programable en dirección Y (Máx. + — 999 de una sola vez).
**Papel:** Rollo de 4,5 pulgadas (114 mm.).

## MONITOR EN COLOR C-1701

**Pantalla:** 13 pulgadas (330 mm.).
**Capacidad de representación:** 25 líneas de 40 caracteres.
**Resolución:** 320 líneas horizontales.
**Compatibilidad:** VIC-20 y COMMODORE 64.
Conectable a un registrador de video.
**Amplificador y altavoz:** Incorporados.




COMPUTER

microelectrónica y control, s. a.

Taquigrafo Serra, 7 5.° Telf. 250 51 03. BARCELONA-29
Princesa, 47 3.° G. Telf. 248 95 70. MADRID-8